Analysis of the Bitcoin Exchange Using Particle MCMC Methods

by

Michael Johnson

M.Sc., University of British Columbia, 2013
B.Sc., University of Winnipeg, 2011

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Statistics and Actuarial Science
Faculty of Science

SIMON FRASER UNIVERSITY
Spring 2017

Approval

Name:	**Michael Johnson**
Degree:	**Master of Science (Statistics)**
Title:	***Analysis of the Bitcoin Exchange Using Particle MCMC Methods***
Examining Committee:	**Chair:** Dr. Jiguo Cao Associate Professor

Dr. Liangliang Wang
Senior Supervisor
Assistant Professor _____

Dr. Dave Campbell
Supervisor
Associate Professor _____

Dr. Tim Swartz
Internal Examiner
Professor _____

Date Defended: March 24, 2017

Abstract

Stochastic volatility models (SVM) are commonly used to model time series data. They have many applications in finance and are useful tools to describe the evolution of asset returns. The motivation for this project is to determine if stochastic volatility models can be used to model Bitcoin exchange rates in a way that can contribute to an effective trading strategy. We consider a basic SVM and several extensions that include fat tails, leverage, and covariate effects. The Bayesian approach with the particle Markov chain Monte Carlo (PMCMC) method is employed to estimate the model parameters. We assess the goodness of the estimated model using the deviance information criterion (DIC). Simulation studies are conducted to assess the performance of particle MCMC and to compare with the traditional MCMC approach. We then apply the proposed method to the Bitcoin exchange rate data and compare the effectiveness of each type of SVM.

Keywords: Stochastic volatility model; hidden Markov model; sequential Monte Carlo; particle Markov chain Monte Carlo; Bitcoin.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Stochastic Volatility Models and Particle Markov Chain Monte Carlo

Stochastic volatility models (SVM) are widely used in the field of economics and finance. They are commonly used for modeling financial time series data such as stock returns and exchange rates [11] or market indices [19]. Applied work in these fields often involves complex nonlinear relationships between many variables and the complicated models used in these situations often give rise to high dimensional integrals that cannot be solved analytically. This has led to the increased popularity of Monte Carlo methods, which can use simulation techniques to estimate complex integrals.

Monte Carlo integration [9] is a simulation technique that uses independent draws from a distribution of interest, referred to as a target distribution, to approximate integrals rather than doing them analytically. However, it is not always possible to sample independent draws directly from the target distribution. In these cases if we can sample slightly dependent draws from a Markov chain, similar approximations can still be made.

A Markov chain is a random or stochastic process that evolves over time from one state to the next. It has the property that the next state depends only on the current state and not on the sequence of states that preceded it. This memoryless characteristic is known as the Markov property. A Markov chain will eventually converge to what is known as a stationary distribution [21]. Markov chain Monte Carlo (MCMC) algorithms create a Markov chain with the target distribution as its stationary distribution. Once the chain has converged the draws are approximately the same as if they were sampled from the target distribution. The two most commonly used MCMC methods are the Gibbs sampler and

the Metropolis-Hastings algorithms [21]. In this project we will make use of both the Gibbs sampler and Metropolis-Hasting methods.

Importance sampling (IS) [9] is another technique which can be used when the target distribution cannot be sampled from directly. Importance sampling involves selecting an importance distribution from which it is easy to draw samples. Then importance weights are calculated and used to re-weight the draws from the important distribution so that they are approximately the same as if they were from the target distribution. If we restrict the importance distribution to a certain form that can be written recursively we can implement sequential importance sampling (SIS) [9] which can be more computationally efficient. The details of these algorithms are outlined in later chapters and we see how they lead into more complicated methods such as sequential Monte Carlo and particle MCMC.

Sequential Monte Carlo (SMC) is an increasingly popular alternative to MCMC, because of its speed and scalability. In general, SMC methods [3] are algorithms that sample from a sequence of target distributions of increasing dimension. The SMC algorithm is basically sequential importance sampling that implements a resampling technique to address some of the problems with SIS. [14] contributed a pioneering paper of SMC focusing on tracking applications. Since then, various SMC algorithms have been proposed to obtain full posterior distributions for problems in nonlinear dynamic systems in science and engineering. To date, these algorithms for nonlinear and non-Gaussian state-space models have been successfully applied in various fields including computer vision, signal processing, tracking, control, econometrics, finance, robotics, and statistics [23, 3, 4, 9]. The SMC algorithm will be explained in more detail in later sections as it is an important part of the particle MCMC algorithm.

Particle Markov chain Monte Carlo (PMCMC) [9] uses sequential Monte Carlo within the MCMC algorithm [2, 15]. As with basic MCMC, both the Gibbs and Metropolis-Hastings methods can be used in PMCMC. [26] used particle Gibbs with ancestor sampling [22] for nontrivial SVMs. In this project the Metropolis-Hastings method is used for PMCMC which is known as the particle marginal Metropolis-Hastings (PMMH) method [2]. The steps of the Particle MCMC algorithm will be explained in detail in a later section.

In this project several types of stochastic volatility models are investigated and the particle Markov chain Monte Carlo method is used to estimate the posterior distributions of the model parameters. We focus on a basic SVM and several extensions that include fat or heavy tails, leverage, and covariate effects. These SVMs have been considered in [26] and [1].

A basic stochastic volatility model was used by [9] to illustrate the effectiveness of the PMCMC method. The basic SVM has uncorrelated Gaussian white noise from the observation

(measurement) equation and the system (state) equation. Many extensions have been proposed to relax the assumption of the uncorrelated error and/or the normal distribution. For example, [17] considered correlated errors and provided an MCMC algorithm for the leverage stochastic volatility model, which extends the basic SVM to accommodate nonzero correlation between Gaussian white noise from the observation equation and the system equation.

Fat-tailed (a standard Student's t distribution), skewed and scale mixture of normal distributions are considered in [24, 5, 13, 17]. Moreover, [10] considered SVMs with jumps. [1] extended a basic SVM to capture a leverage effect, a fat-tailed distribution of asset returns and a nonlinear relationship between the current volatility and the previous volatility process. The author used the Bayesian approach with the MCMC method to estimate model parameters, and evaluate different models with several Bayesian model selection criteria.

Besides the univariate SVMs, [16, 29] focused on the multivariate stochastic volatility models. [6] generalizes the popular stochastic volatility in mean model of [18] to allow for time-varying parameters in the conditional mean.

Since several different SVMs will be investigated it is important that we have a way to assess their effectiveness and select the best estimated model for a given application. For example, [17] applied several Bayesian model selection criteria that include the Bayes factor, the Bayesian predictive information criterion and the deviance information criterion.

1.2 Bitcoin and Bitcoin Exchanges

Bitcoin is an electronic payment system that has been growing in popularity over the last several years. It was first introduced by Satoshi Nakamoto who published the white paper [25] in 2008 and released it as open-source software in 2009. Bitcoin is a type of cryptocurrency which is defined as an "electronic payment system based on cryptographic proof" [25]. The system allows transactions to take place directly between users, without a central payment system or any single administrator. In addition Bitcoins are not linked to any type of commodity such as gold or silver [27]. Therefore this decentralized virtual currency is controlled by its users instead of a governing body. The Bitcoin system utilizes a peer-to-peer network of all those who are involved in creating and trading Bitcoins, to process and check all transactions.

Today Bitcoins are accepted as payment for goods and services by many online e-commerce sites and by an increasing number of physical stores by way of smart phone apps. Bitcoin transactions are attractive to merchants due to their high speed and low transaction

fees. Simon Fraser University has recently began accepting tuition payment in Bitcoin and has introduced a Bitcoin ATM in the bookstore. Although Bitcoins are becoming more mainstream, the concept of virtual money can be confusing at first glance to the average consumer. This section will explain the basics of Bitcoins and where they come from as well as how they are bought and sold on online exchanges.

To begin, Bitcoins are created though a process know as mining. The basic idea is that users offer their computing power for payment processing work and they are rewarded with Bitcoins. Bitcoins are exchanged over the network all the time and these transactions must be verified and recorded. A list of all the transactions made during a set period of time is called a block. Every Bitcoin transaction ever made is recorded in a public ledger made up of a long list of blocks, known as the Blockchain. When a new block of transactions is created it is the miners job to put it through a confirmation process and then add it to the Blockchain. This process is very computationally expensive as it requires finding solutions to complex mathematical problems. Miners use software and machines specifically designed for Bitcoin mining and are rewarded with new Bitcoins every time the Blockchain is updated. Bitcoin mining is a complicated process and many people do not have the means to acquire Bitcoins in this way. However, it is possible to simply purchase Bitcoins with traditional currency from miners or anyone that is looking to sell them.

Bitcoin is traded on many online exchanges where it can be bought or sold using regular government backed currencies. There are exchanges that accept Canadian Dollars (CND), Chinese Yuan (CNY) and US Dollars (USD). Exchanges such as OKCoin, BitStamp, or Kraken allow users to deposit and withdraw funds from the exchange via regular online banking services. Kraken is currently the largest exchange to accept Canadian Dollars. In addition to Bitcoin, many types of similar cryptocurrencies exists today such as Litecoin or Ethereum which have been growing in popularity, but bitcoin is by far the most widely used and largest in terms of total market value. Many Bitcoin exchanges allow users to purchase other forms of cryptocurrencies as well, or to trade Bitcoin directly for other cryptocurrencies. This project focuses specifically on Bitcoin trading, but the exchange rate of other cryptocurrencies will likely follow a similar pattern and could be an interesting future application of this work.

Bitcoin exchange rates can be extremely volatile and an effective trading strategy could potentially lead to large profits. The value of Bitcoin may not behave like a typical currency. Econonomic and financial theory cannot explain the large volatility in the Bitcoin price. Factors such as interest rates and inflation do not effect Bitcoin as they would a government backed currency because there is no central bank overseeing the issuing of Bitcoin. Therefore, Bitcoin price is "driven solely by the investors faith in the perpetual growth" [20]. A statistical analysis of the log-returns of the exchange rate of Bitcoin in US dollars

was provided by [7]. Parametric distributions that are popular in financial applications are fitted to the log-returns and the generalized hyperbolic distribution is shown to give the best fit. The links between Bitcoin price and socials signals was examined by [8]. Using data from Bitcoin exchanges, social media and Google search trends they found evidence of positive feedback loops. An increase in popularity of Bitcoin leads to an increase in Google searches for Bitcoin and social media coverage. However, their results failed to explain sudden negative changes in Bitcoin price. In this project we attempt to use SVMs from the field of finance and economics to model the exchange rate of Bitcoin. The PMCMC method will be used to estimate the SVM parameters.

1.3 Research Objective

This project begins with an examination of the use of stochastic volatility models in financial applications. We attempt to use several types of stochastic volatility models to describe the evolution of the exchange rate of Bitcoin. We consider a basic stochastic volatility model and several extensions that include heavy tails, leverage, and covariates. The Bayesian approach with the particle Markov chain Monte Carlo (PMCMC) method is employed to estimate the model parameters. Simulation studies are conducted to assess the performance of particle MCMC and to compare with the traditional MCMC approach. We then apply the proposed method to the Bitcoin exchange rate data.

This project is focused on particle Markov chain Monte Carlo and the application of modeling Bitcoin exchange rates. Therefore the main research objectives are:

i Conduct simulation studies to evaluate the performance of PMCMC.

ii Explore several SVMs for modeling the Bitcoin exchange rate and estimate the model parameters using the proposed PMCMC method.

iii Select the most appropriate model for the Bitcoin application.

This project was motivated by a desire to understand the Bitcoin market and an interest in the extremely challenging problem of modeling and forecasting financial markets. The ultimate goal for this research is to find a way to model Bitcoin exchange rates in a way that can contribute to an effective trading strategy.

1.4 Thesis Organization

The rest of the thesis is organized as follows. A description of the stochastic volatility models used in this project are given in Chapter 2. A detailed description of the MCMC and PMCMC algorithms are presented in Chapter 3. Simulation studies are conducted and the performance of particle MCMC and the traditional MCMC approach are compared in Chapter 4. The proposed methods are applied to real Bitcoin exchange rate data in Chapter 5. Chapter 6 provides concluding remarks.

Chapter 2

Stochastic Volatility Models

2.1 Hidden Markov Models

A Markov process is a stochastic process where the future states depend only on the current state and not on the sequence of states that preceded it. This memoryless characteristic is known as the Markov property. In a hidden Markov model with unknown parameters θ, the underlying (hidden) process $X_{1:n}$ is assumed to be a Markov process with initial distribution $\mu_\theta(x_1)$ and transitional distribution $f_\theta(x_t|x_{t-1})$. The observations $Y_{1:n}$ are assumed to be conditionally independent given the process $X_{1:n}$ and have the marginal distribution $g_\theta(y_t|x_t)$. Figure 2.1 illustrates how the unobserved underlying process $X_{1:n}$ relates to the observed values $Y_{1:n}$.

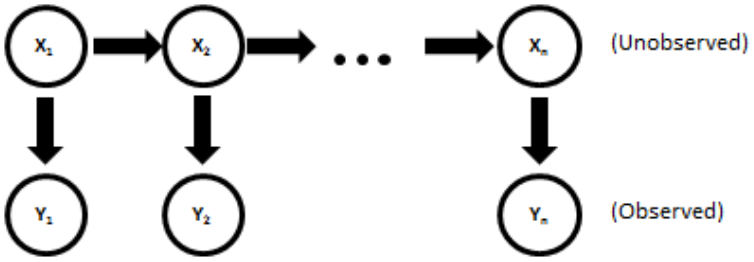

Figure 2.1: Illustration of a Hidden Markov Model.

In summary, a hidden Markov model is described as follows:

$$X_1 \sim \mu_\theta(x_1), \tag{2.1}$$

$$(X_t|X_{t-1} = x_{t-1}) \sim f_\theta(x_t|x_{t-1}), \tag{2.2}$$

$$(Y_t|X_t = x_t) \sim g_\theta(y_t|x_t), \tag{2.3}$$

where $X_t \in \mathcal{X}$ and $Y_t \in \mathcal{Y}$. In the following sections we will consider several different stochastic volatility models that can be written in the form of hidden Markov models.

2.2 Basic Stochastic Volatility Model

2.2.1 Basic SVM Version 1

First we will consider the following basic stochastic volatility model [9], with observations $Y_{1:n}$ and underlying process $X_{1:n}$. We have $\mathcal{X} = \mathcal{Y} = \mathbb{R}$,

$$X_t = \alpha \cdot X_{t-1} + \sigma \cdot V_t,$$

$$Y_t = \beta \cdot \exp\{X_t/2\} \cdot U_t,$$

where $X_1 \sim N(0, \frac{\sigma^2}{1-\alpha^2})$, $V_t \sim N(0,1)$, $U_t \sim N(0,1)$ and $\theta = (\alpha, \beta, \sigma^2)$ is unknown. Here, U_t and V_t are uncorrelated Gaussian white noise sequences. The scaling factor $\exp(X_t/2)$ specifies the amount of volatility at time t, σ determines the volatility or log-volatility and α measures the autocorrelation [9].

Recall that if $Z \sim N(0,1)$, then $(a + bZ) \sim N(a, b^2)$. Therefore, the model can be described as follows:

$$\mu_\theta(x_1) = N\left[x_1; 0, \frac{\sigma^2}{1-\alpha^2}\right],$$

$$f_\theta(x_t|x_{t-1}) = N\left[x_t; \alpha x_{t-1}, \sigma^2\right],$$

$$g_\theta(y_t|x_t) = N\left[y_t; 0, \beta^2 \exp(x_t)\right].$$

In the next section we will see another version of this basic stochastic volatility model that will also be used in this project.

2.2.2 Basic SVM Version 2

Consider the following alternative parameterization of the basic stochastic volatility model, with observations $Y_{1:n}$ and underlying process $X_{1:n}$ [26]. We have $\mathcal{X} = \mathcal{Y} = \mathbb{R}$,

$$X_t = \mu_x + \alpha \cdot (X_{t-1} - \mu_x) + \sigma \cdot V_t,$$

$$Y_t = \mu_y + \sqrt{\gamma} \cdot \exp\{X_t/2\} \cdot U_t,$$

where $X_1 \sim N(\mu_x, \frac{\sigma^2}{1-\alpha^2})$, $V_t \sim N(0,1)$, $U_t \sim N(0,1)$ and $\theta = (\alpha, \mu_x, \mu_y, \sigma^2)$ is unknown. Here U_t and V_t are uncorrelated Gaussian white noise sequences, and μ_x is the drift term in the state equation. The scaling factor $\exp(X_t/2)$ specifies the amount of volatility at time t, σ determines the volatility or log-volatility and α is the persistence parameter that measures the autocorrelation. We impose that $|\alpha| < 1$ such that we have a stationary process with the initial distribution $\mu(x_1) = N(x_1; \mu_x, \frac{\sigma^2}{1-\alpha^2})$. To ensure identifiability, we fix γ to be 1, and leave μ_x unrestricted [26]. This model is just a re-parameterization of the basic model in Section 2.2.1, and if we fix $\mu_x = 0$ and $\mu_y = 0$, we can see that it is in the same form as the previous section.

In this case, the model can be described as follows:

$$\mu_\theta(x_1) = N\left[x_1; \mu_x, \frac{\sigma^2}{1-\alpha^2}\right],$$

$$f_\theta(x_t|x_{t-1}) = N\left[x_t; \mu_x + \alpha(x_{t-1} - \mu_x), \sigma^2\right],$$

$$g_\theta(y_t|x_t) = N\left[y; \mu_y, \exp(x_t)\right].$$

The basic stochastic volatility model can be too restrictive for many financial time series [26]. In the following subsections, we will consider several extension of the basic stochastic volatility model.

2.3 Stochastic Volatility Model with Fat Tails

Consider the following stochastic volatility model with fat tails (heavy tails) [5, 13, 17]. We have observations $Y_{1:n}$ and underlying process $X_{1:n}$ with $\mathcal{X} = \mathcal{Y} = \mathbb{R}$. The model is defined

as follows:

$$X_t = \mu_x + \alpha \cdot (X_{t-1} - \mu_x) + \sigma \cdot V_t,$$

$$Y_t = \mu_y + \exp\{X_t/2\} \cdot U_t,$$

where $U_t \sim t_\nu$ and $\theta = (\alpha, \mu_x, \mu_y, \nu, \sigma^2)$ is unknown. Here t_ν denotes a Student-t distribution with $\nu > 2$ degrees of freedom. This model can be described as follows:

$$\mu_\theta(x_1) = N\left[x_1; \mu_x, \frac{\sigma^2}{1-\alpha^2}\right],$$

$$f_\theta(x_t|x_{t-1}) = N\left[x_t; \mu_x + \alpha(x_{t-1} - \mu_x), \sigma^2\right],$$

$$g_\theta(y_t|x_t) = t_\nu\left[y; \mu_y, \exp(x_t), \nu\right].$$

The stochastic volatility model with fat tails can accommodate a wide range of kurtosis and is particularly important when dealing with extreme observations or outliers [24].

2.4 Stochastic Volatility Model with Leverage Effect

Consider the following stochastic volatility model with leverage effect [26]. We have observations $Y_{1:n}$ and underlying process $X_{1:n}$ with $\mathcal{X} = \mathcal{Y} = \mathbb{R}$. The model is defined as follows:

$$X_t = \mu_x + \alpha \cdot (X_{t-1} - \mu_x) + \sigma \cdot V_t,$$

$$Y_t = \mu_y + \exp\{X_t/2\} \cdot U_t,$$

where U_t and V_t are correlated. We write $U_t = \rho V_t + \sqrt{(1-\rho^2)} \cdot \xi_t$, where $\xi_t \sim N(0,1)$, and they are uncorrelated with V_t. In this way, $Y_t|V_t \sim N\left(\breve{\mu}_t, \breve{\sigma}_t^2\right)$, where $\breve{\mu}_t = \rho \cdot \exp(x_t/2) \cdot V_t$, $\breve{\sigma}_t^2 = (1-\rho^2) \cdot \exp(x_t)$ and X_t follows the one in the basic stochastic volatility model. We have $V_t = \sigma^{-1} \cdot [(X_t - \mu_x) - \alpha \cdot (X_{t-1} - \mu_x)]$.

In order to use similar algorithms for HMM to estimate the hidden process, we define the state as $\mathbf{X}_t = (X_{t+1}, X_t)^T$. In other words, the stochastic volatility model with leverage can be expressed in the form of a non-linear, non-Gaussian state space model with

$$g_\theta(y_t|\mathbf{x}_t) = N\left(y; \rho \cdot \exp(x_t/2) \cdot \sigma^{-1} \cdot [(x_t - \mu_x) - \alpha \cdot (x_{t-1} - \mu_x)], (1-\rho^2) \cdot \exp(x_t)\right).$$

The state transition function is:

$$\begin{pmatrix} X_{t+1} \\ X_t \end{pmatrix} = \begin{pmatrix} \mu_x(1-\alpha) \\ 0 \end{pmatrix} + \begin{pmatrix} \alpha & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} X_t \\ X_{t-1} \end{pmatrix} + \begin{pmatrix} V_t \\ 0 \end{pmatrix}.$$

That is,

$$f_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}) = N_2(\mathbf{x}_t; \boldsymbol{\mu}_x + A_x\mathbf{x}_{t-1}, \Sigma_x),$$

where $\boldsymbol{\mu}_x = (\mu_x(1-\alpha), 0)^T$, $A_x = \begin{pmatrix} \alpha & 0 \\ 1 & 0 \end{pmatrix}$, and $\Sigma_x = \begin{pmatrix} \sigma^2 & 0 \\ 0 & 0 \end{pmatrix}$.

2.5 Stochastic Volatility Model with Covariate Effects

Finally we consider the following stochastic volatility model which allows for covariate effects. We have observations $\{Y_n\}$ and underlying process $\{X_n\}$ with $\mathcal{X} = \mathcal{Y} = \mathbb{R}$. The model is defined as follows:

$$\begin{aligned} X_t &= \mu_x + \alpha\cdot(X_{t-1} - \mu_x) + \sigma\cdot V_t, \\ Y_t &= W_t^{'}\eta + \exp\{X_t/2\}\cdot U_t, \end{aligned}$$

where W_t is a $q \times 1$ vector of covariates, η is the associated $q \times 1$ vector of parameters and $\theta = (\alpha, \mu_x, \eta, \sigma^2)$. This model can be described by the following functions:

$$\mu_\theta(x_1) = N\left(x_1; \mu_x, \frac{\sigma^2}{1-\alpha^2}\right),$$

$$f_\theta(x_t|x_{t-1}) = N\left(x_t; \mu_x + \alpha(x_{t-1} - \mu_x), \sigma^2\right),$$

$$g_\theta(y_t|x_t) = N\left(y; W_t^{'}\eta, \exp(x_t)\right).$$

2.6 Chapter Summary

In this section the concept of hidden Markov models was introduced and we looked at several types of stochastic volatility models that can be expressed in this form. We considered a basic SVM and several extensions that included heavy tails, leverage and covariate effect. Particle Markov chain Monte Carlo algorithms will be applied to each of these models to estimate the parameter values. The models will be fit to simulated data sets and Bitcoin

exchange rate data. In the next section we will outline the details of the particle MCMC method and show how it can be used to estimate the model parameters.

Chapter 3

Bayesian Inference for Stochastic Volatility Models

3.1 Bayesian Inference

Consider a stochastic volatility model with hidden process $X_{1:n}$, observations $Y_{1:n}$ and a fixed vector of parameters θ. In the Bayesian framework, equations (2.1) and (2.2) define the prior distribution of the hidden process as follows:

$$p_\theta(x_{1:n}) = \mu_\theta(x_1) \prod_{t=2}^{n} f_\theta(x_t|x_{t-1}),$$

and equation (2.3) defines the following likelihood function,

$$p_\theta(y_{1:n}|x_{1:n}) = \prod_{t=1}^{n} g_\theta(y_t|x_t).$$

Consequently, given θ, the posterior distribution of $X_{1:n}$ given the observed data $Y_{1:n}$ is:

$$p_\theta(x_{1:n}|y_{1:n}) = \frac{p_\theta(x_{1:n}, y_{1:n})}{p_\theta(y_{1:n})}, \tag{3.1}$$

where

$$p_\theta(x_{1:n}, y_{1:n}) = p_\theta(x_{1:n}) p_\theta(y_{1:n}|x_{1:n}) = \mu_\theta(x_1) \prod_{t=2}^{n} f_\theta(x_t|x_{t-1}) \prod_{t=1}^{n} g_\theta(y_t|x_t),$$

and

$$p_\theta(y_{1:n}) = \int p_\theta(x_{1:n}, y_{1:n}) dx_{1:n}. \tag{3.2}$$

When the parameters θ are unknown, the posterior distribution of θ and $X_{1:n}$ is:

$$p(\theta, x_{1:n}|y_{1:n}) \propto p_\theta(x_{1:n}, y_{1:n}) \cdot p(\theta),$$

where $p(\theta)$ is the prior for θ.

For the simplest cases, the finite state-space hidden Markov models, the integral in equation (3.2) can be computed exactly. For linear Gaussian models, the posterior distribution $p_\theta(x_{1:n}|y_{1:n})$ is also a Gaussian distribution whose mean and covariance can be computed using the Kalman Filter [9]. However, it is impossible to compute the integral in equation (3.2) in a closed form for most non-linear non-Gaussian models. Unfortunately, the stochastic volatility models of interest in this project belong to the latter case and we have to use numerical approximations.

3.2 Monte Carlo Integration

Monte Carlo Integration is a simulation technique that uses independent draws from a distribution to approximate integrals rather than solving them analytically. The distribution that is being approximated is known as the target distribution, denoted as $\pi_n(x_{1:n})$. In our case the target distribution is the posterior, so $\pi_n(x_{1:n}) = p_\theta(x_{1:n}|y_{1:n})$. The Monte Carlo method involves sampling N independent draws, $X^k_{1:n} \sim \pi_n(x_{1:n})$, $k = 1, \ldots, N$, and then approximating $\pi_n(x_{1:n})$ by the empirical measure,

$$\widehat{\pi}_n(x_{1:n}) = \frac{1}{N} \sum_{k=1}^{N} \delta_{X^k_{1:n}}(x_{1:n}),$$

where $\delta_{x_0}(x)$ denotes the Dirac delta mass located at x_0 [3]. Consider the integral

$$I = \int m(x_{1:n}) \pi_n(x_{1:n}) dx_{1:n},$$

where $m(x_{1:n})$ is some function of $x_{1:n}$. Then the integral I can be approximated using Monte Carlo integration by

$$\widehat{I} = \frac{1}{N} \sum_{k=1}^{N} m(X^k_{1:n}). \tag{3.3}$$

By the strong law of large numbers $\widehat{I} \longrightarrow I$ as $N \longrightarrow \infty$ [21]. However, in some cases it is not possible to generate independent draws from the target distribution. For example, suppose we want to sample from the posterior distribution $p_\theta(x_{1:n}|y_{1:n})$, but the normalizing constant $p_\theta(y_{1:n})$ is unknown. If we can sample slightly dependent draws from the posterior

distribution using a Markov chain, then it is still possible to estimate the integrals or quantities of interest using equation (3.3).

3.3 Posterior Inference via Markov Chain Monte Carlo

3.3.1 Markov Chain Monte Carlo (MCMC)

A Markov chain is a stochastic process where the future states depend only on the current state and not on the sequence of states that preceded it. For example, let z_t be the state of a stochastic process at time t. Then, the future state z_{t+1} depends only on the current state z_t. In other words, a stochastic process is considered a Markov chain if it satisfies the Markov property:

$$p(z_{t+1}|z_1, z_2, ..., z_t) = p(z_{t+1}|z_t). \tag{3.4}$$

Under mild conditions a Markov chain will eventually converge to what is known as a stationary or limiting distribution [21]. If we can create a Markov chain whose stationary distribution is the targeted posterior distribution $p_\theta(x_{1:n}|y_{1:n})$, then the chain can be run to get draws that are approximately from $p_\theta(x_{1:n}|y_{1:n})$ once it has converged.

A Markov chain should converge to the desired stationary distribution regardless of the starting point, however the time it takes to converge will vary [21]. Therefore, it is common practice to discard a certain number of the first draws, a process known as a burn-in. This helps to assure that the draws are closer to the stationary distribution and less dependent on the starting point.

Once the Markov chain has converged the draws will be approximately the same as if they were drawn from the posterior $p_\theta(x_{1:n}|y_{1:n})$. However, these draws will not be independent, which is required for Monte Carlo Integration. Fortunately, the Ergodic Theorem allows the dependence between draws of the Markov chain to be ignored [21].

In summary MCMC is a simulation technique that involves taking draws from a Markov chain that has the desired posterior distribution as its stationary distribution. In general, there are two MCMC algorithms that are most commonly used: the Gibbs sampler and the Metropolis-Hastings algorithm. In the next section, we will outline the Gibbs sampling method used in this project.

3.3.2 Gibbs Sampling

For this project a basic MCMC algorithm using the Gibbs sampling method was implemented as a comparison to PMCMC. Consider the basic stochastic volatility model from the Section 2.2.1, a Gibbs MCMC algorithm can be used to estimate the model parameters $\theta = (\alpha, \beta, \sigma)$. Let $t = 1, ..., n$ be the number of time steps, let m be the number of iterations and $\theta^{(i)}$ be the parameters values at the i^{th} MCMC iteration $(i = 1, ..., m)$.

First starting values must be selected the model parameters α, β and σ. The prior distributions for the parameters are: $x_t \sim N(0, 1)$, $\alpha \sim Uniform(-1, 1)$, $\beta^2 \sim IG(\nu_0/2, \gamma_0/2)$ and $\sigma^2 \sim IG(\nu_0/2, \gamma_0/2)$, where $IG(\cdot, \cdot)$ is the inverse Gamma distribution with the shape and rate parameters. Then the posterior distribution of interest is:

$$\begin{aligned}\pi(\alpha, \beta^2, \sigma^2, x_{1:n}|y_{1:n}) \propto & \prod_{t=1}^{n} \left[\phi(y_t; 0, \beta^2 \cdot exp(x_t))\right] \cdot \phi(x_1; 0, \sigma^2) \\ & \cdot \prod_{t=2}^{n} \left[\phi(x_t; \alpha x_{t-1}, \sigma^2)\right] \cdot I[\alpha\epsilon(-1, 1)] \\ & \cdot IG(\sigma^2; \nu_0/2, \gamma_0/2) \cdot IG(\beta^2; \nu_0/2, \gamma_0/2).\end{aligned}$$

In order to implement Gibbs sampling we must first calculate the full conditional distributions for each parameter. A full conditional distribution is defined as the distribution of a parameter conditional on the known information and all other parameters. The full conditional distributions for this model are as follows:

$$\alpha|\cdot \sim N\left[\frac{\sum_{t=2}^{n} x_{t-1}x_t}{\sum_{t=2}^{n} x_{t-1}^2}, \frac{\sigma^2}{\sum_{t=2}^{n} x_{t-1}^2}\right] \cdot I[\alpha\epsilon(-1, 1)], \tag{3.5}$$

$$\beta^2|\cdot \sim IG\left[\frac{n + \nu_0}{2}, \frac{1}{2}\left(\gamma_0 + \sum_{t=1}^{n} \frac{y_t^2}{\exp(t_n)}\right)\right], \tag{3.6}$$

$$\sigma^2|\cdot \sim IG\left[\frac{n + \nu_0}{2}, \frac{1}{2}\left(x_1^2 + \gamma_0 + \sum_{t=2}^{n}(x_t - \alpha x_{t-1})^2\right)\right]. \tag{3.7}$$

Then, assuming $x_0 = 0$, the density of the full conditional distribution for x_t is:

$$p(x_t|\cdot) \propto \phi\left(x_t; \alpha\frac{(x_{t-1} + x_{t+1})}{1 + \alpha^2}, \frac{\sigma^2}{1 + \alpha^2}\right) \cdot \left(\frac{1}{\beta^2 \exp(x_t)}\right)^{1/2} \exp\left(\frac{-y_t^2}{2\beta^2 \exp(x_t)}\right), \tag{3.8}$$

for $t = 1, 2, ..., n - 1$.
At t=n,

$$p(x_t|\cdot) \propto \phi(x_t; \alpha x_{t-1}, \sigma^2) \cdot \phi(y_t; 0, \beta^2 \exp(x_t)). \tag{3.9}$$

However, this distribution is non-standard and cannot be sampled from directly. Therefore, $x_t|\cdot$ is proposed using an accept-reject sampler by sampling from:

$$\begin{aligned} q(x_t) &\propto \phi\left[x_t; m_t, \sigma_t^2\right] \cdot g^*(y_t|x_t, \beta) \\ &\propto \phi\left[x_t; m_t, \sigma_t^2\right] \cdot \exp\left[\frac{-x_t}{2} + \frac{y_t^2}{2\beta^2} \cdot x_t \cdot \exp(m_t)\right] \\ &\propto N\left[x_t; m_t + \frac{\sigma_t^2}{2}\left(\frac{y_t^2}{\beta^2} \cdot \exp(-m_t) - 1\right), \sigma_t^2\right], \end{aligned}$$

where $m_t = \dfrac{\alpha(x_{t-1} + x_{t+1})}{1+\sigma^2}$ and $\sigma_t^2 = \dfrac{\sigma^2}{1+\sigma^2}$ for $t = 1, 2, ..., n-1$, $m_t = \alpha \cdot x_{t-1}$ and $\sigma_t^2 = \sigma^2$ for $t = n$, and $g^*(y_t|x_t, \beta) \propto \exp\left[-\dfrac{x_t}{2} + \dfrac{y_t^2}{2\beta^2} \cdot x_t \cdot \exp(m_t)\right]$.

At each iteration, we propose x_t from $q(x_t)$ until one of these is accepted with probability:

$$\frac{g(y_t|x_t, \beta)}{g^*(y_t|x_t, \beta)},$$

where $g(y_t|x_t, \beta) = \phi\left[y_t; 0, \beta^2 \exp(x_t)\right]$. The steps of the Gibbs sampling algorithm are outlined in Table 3.1.

Table 3.1: Gibbs Sampler Algorithm.

Draw initial parameter values $x_{1:n}^{(0)}$ and $\theta^{(0)} = \left(\alpha^{(0)}, \beta^{2(0)}, \sigma^{2(0)}\right)$ from their prior distributions.

For $i = 1, ..., m$:

- Draw $\alpha^{(i)}$ from its full conditional using equation (3.5).
- Draw $\beta^{2(i)}$ from its full conditional using equation (3.6).
- Draw $\sigma^{2(i)}$ from its full conditional using equation (3.7).
- Update $x_{1:n}^{(i)}$ using the accept-reject sampler.

In cases it is not possible to calculate all the full conditionals necessary to implement the Gibbs sampling and a different method such as the Metropolis-Hastings algorithm or the accept-reject sampler must be used. A Gibbs sampling algorithm is also very model specific

because the full conditionals must be recalculated for different models. For this reason an MCMC algorithm with Gibbs sampling was only developed for the basic stochastic volatility model.

3.4 Posterior Inference via Sequential Monte Carlo (SMC)

The purpose of this section is to introduce sequential Monte Carlo methods which are an important part of the particle MCMC algorithm. This section focuses on estimating the hidden process $X_{1:n}$ and it is assumed that the parameters, θ are fixed. We omit θ from the general notation for simplicity.

3.4.1 Importance Sampling (IS)

As previously mentioned a problem with Monte Carlo integration is that it might not be possible to sample directly from the target distribution. Importance sampling (IS) is another technique to address this problem. Let $\pi_n(x_{1:n})$ be the target distribution and $\gamma_n(x_{1:n})$ be the unnormalized target distribution. Then

$$\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n},$$

where $Z_n = \int \gamma_n(x_{1:n})\, dx_{1:n}$. Recall, that in this case the posterior $p_\theta(x_{1:n}|y_{1:n})$ is our target distribution. Therefore, we have:

$$\pi_n(x_{1:n}) = p_\theta(x_{1:n}|y_{1:n}) = \frac{p_\theta(x_{1:n}, y_{1:n})}{p_\theta(y_{1:n})} = \frac{p_\theta(y_{1:n}|x_{1:n}) \cdot p_\theta(x_{1:n})}{p_\theta(y_{1:n})}.$$

In order to implement importance sampling we must select an importance distribution $q_n(x_{1:n})$ from which it is easy to draw samples and use the following IS identities.

$$\pi_n(x_{1:n}) = \frac{w_n(x_{1:n}) q_n(x_{1:n})}{Z_n}, \tag{3.10}$$

where

$$Z_n = \int w_n(x_{1:n}) q_n(x_{1:n}) dx_{1:n}, \tag{3.11}$$

and the unnormalized weight function $w_n(x_{1:n})$ is defined as

$$w_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})}.$$

Then draw N independent samples $X_{1:n}^k \sim q_n(x_{1:n})$, which are commonly referred to as particles, and use the Monte Carlo approximation of $q_n(x_{1:n})$ in equations 3.10 and 3.11 to obtain the following estimate for the target distribution

$$\widehat{\pi}_n(x_{1:n}) = \sum_{k=1}^{N} W_n^k \delta_{X_{1:n}^k}(x_{1:n}), \tag{3.12}$$

$$\widehat{Z}_n = \frac{1}{N} \sum_{i=k}^{N} w_n(X_{1:n}^k), \tag{3.13}$$

where W_n^k are the normalized weights, which are defined as

$$W_n^k = \frac{w_n(X_{1:n}^k)}{\sum_{j=1}^{N} w_n(X_{1:n}^j)}.$$

In summary, the basic idea of importance sampling is to draw samples from the importance distribution and re-weight them using the importance weights to approximate the target distribution.

3.4.2 Sequential Importance Sampling (SIS)

Another problem with Monte Carlo methods is that even if it is possible to sample from the target distribution the computational complexity increases at least linearly with n. This problem can be addressed by using sequential importance sampling (SIS) which has a fixed computational complexity at each time step [9]. Sequential importance sampling is a special case of importance sampling where the importance distribution, $q_n(x_{1:n})$ must be of the following form:

$$\begin{aligned} q_n(x_{1:n}) &= q_{n-1}(x_{1:n-1}) q_n(x_n | x_{1:n-1}) \\ &= q_1(x_1) \prod_{t=2}^{n} q_t(x_t | x_{1:t-1}). \end{aligned}$$

In order to obtain N draws $X_{1:n}^i \sim q_n(x_{1:n})$ at time n, sample $X_1^i \sim q_1(x_1)$ at time 1, then sample $X_k^i \sim q_k(x_k | X_{1:k-1}^i)$ at time k for $k = 2, ..., n$. The unnormalized weights are computed recursively as:

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})} \\ &= \frac{\gamma_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \cdot \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1}) q_n(x_n | x_{1:n-1})}, \end{aligned}$$

which can be written in the form:

$$\begin{aligned} w_n(x_{1:n}) &= w_{n-1}(x_{1:n-1}) \cdot \alpha_n(x_{1:n}) \\ &= w_1(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:k}), \end{aligned}$$

where $\alpha_n(x_{1:n})$ is the incremental importance weight and is given by

$$\alpha_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1}) q_n(x_n|x_{1:n-1})}.$$

In the case of hidden Markov Models $w_n(x_{1:n})$ can be simplified by selecting an importance distribution such that

$$\begin{aligned} q(x_n|x_{1:n-1}) &= p_\theta(x_n|x_{n-1}) = f(x_n|x_{n-1}), \\ q_n(x_{1:n}) &= p_\theta(x_{1:n}) = \mu(x_1) \prod_{k=2}^{n} f(x_k|x_{k-1}). \end{aligned}$$

Then $w_n(x_n)$ simplifies to

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\gamma_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \cdot \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1}) q_n(x_n|x_{1:n-1})} \\ &= \frac{p(x_{1:n-1}) p(y_{1:n-1}|x_{1:n-1})}{p(x_{1:n-1})} \cdot \frac{p(x_{1:n}) p(y_{1:n}|x_{1:n})}{p(x_n|x_{n-1}) p(x_{1:n-1}) p(y_{1:n-1}|x_{1:n-1})} \\ &= \prod_{k=1}^{n-1} g(y_k|x_k) \cdot \frac{\mu(x_1) \cdot \prod_{k=2}^{n} f(x_k|x_{k-1}) \cdot \prod_{k=1}^{n} g(y_k|x_k)}{\left[\mu(x_1) \cdot \prod_{k=2}^{n-1} f(x_k|x_{k-1}) \cdot \prod_{k=1}^{n-1} g(y_k|x_k) \right] \cdot f(x_n|x_{n-1})} \\ &= \prod_{k=1}^{n-1} g(y_k|x_k) \cdot g(y_n|x_n) \\ w_n(x_{1:n}) &= w_{n-1}(x_{1:n-1}) \cdot g(y_n|x_n), \end{aligned}$$

where $\alpha_n(x_{1:n}) = g(y_n|x_n)$ is the incremental importance weight.

The sequential importance sampling algorithm is outlined in Table 3.2. At any time n we can compute the estimates $\widehat{\pi}_n(x_{1:n})$ and $\widehat{Z}_n$ from the equations 3.12 and 3.13, respectively.

Table 3.2: Sequential Importance Sampling Algorithm.

At time $t = 1$

For $k = 1, ..., N$:

- Sample $X_1^k \sim q_1(x_1)$.
- Set the unnormalized weights to $w_1(X_1^k) = g(y_1|X_1^k)$.

- Compute the normalized weights: $W_1^k = \dfrac{w_1(X_1^k)}{\sum_{j=1}^N w_1(X_1^j)}$.

At time $t = 2, ..., n$

For $k = 1, ..., N$:

- Sample $X_t^k \sim q_t(x_t|X_{1:t-1}^k)$.
- Compute the unnormalized weights: $w_t(X_{1:t}^k) = w_{t-1}(X_{1:t-1}^k) \cdot g(y_t|X_t^k)$.

- Compute the normalized weights: $W_t^k = \dfrac{w_t(X_{1:t}^k)}{\sum_{j=1}^N w_t(X_{1:t}^j)}$.

A sensibly chosen importance distribution will allow the time required to sample from $q_n(x_n|x_{1:n-1})$ and to compute $\alpha_n(x_{1:n})$ to be independent of n [9]. However, the variance of the estimates increase exponentially with n, which is a major drawback of the SIS method [9].

3.4.3 Sequential Monte Carlo (SMC)

Sequential Monte Carlo is essentially sequential importance sampling that implements a resampling technique to address the problem of the increasing variance of the estimates. Resampling refers to sampling from an approximation which was itself obtained by sampling [9]. In this case we are resampling from the SIS approximation $\widehat{\pi}_n(x_{1:n})$ which is equivalent to selecting $X_{1:n}^k$ with probability W_n^k. The SMC algorithm is very similar to SIS except that resampling is performed at each time step. The resampling step leads to a high probability of removing the particles with low weights. In the sequential framework this means that particles with low weights are not carried forward and computational efforts can be focused on regions with high probability mass [9]. The SMC algorithm is summarized in table 3.3.

Table 3.3: Sequential Monte Carlo Algorithm.

At time $t = 1$

For $k = 1, ..., N$:

- Draw $X_1^k \sim \mu(x_1^k)$.
- Set $w_1(X_1^k) = g(y_1|X_1^k)$.

- Normalize the importance weights: $W_1^k = \dfrac{w_1(X_1^k)}{\sum_{j=1}^N w_1(X_1^j)}$.

At time $t = 2, ..., n$

- Resample N particles with probabilities $\left\{W_{1:t-1}^k\right\}_{k=1}^N$ and for $k = 1, ..., N$ set $w_{t-1}(X_{1:t-1}^k) = \frac{1}{N}$.

For $k = 1, ..., N$:

- Draw $X_t^k \sim f(X_t^k|X_{t-1}^k)$.
- Compute the importance weights: $w_t(X_{1:t}^k) = w_{t-1}(X_{1:t-1}^k) \cdot g(y_t|X_t^k)$.

- Normalize the importance weights: $W_{1:t}^k = \dfrac{w_t(X_{1:t}^k)}{\sum_{j=1}^N w_1(X_{1:t}^j)}$.

The methods discussed in this section will only provide an approximation for the hidden process X_n in a stochastic volatility model. In the next section we will see how to estimate the model parameters, θ, using particle Markov chain Monte Carlo.

3.5 Particle Markov Chain Monte Carlo (PMCMC)

Particle Markov chain Monte Carlo (PMCMC) uses the sequential Monte Carlo method within the MCMC algorithm. As with basic MCMC, both the Gibbs and Metropolis-Hastings methods can be used in PMCMC. In this project we use the particle marginal Metropolis-Hasting (PMMH) method [2]. Our goal is to provide an estimate for the stochastic volatility model parameters, θ, and the posterior distribution, $p_\theta(x_{1:n}|y_{1:n})$.

Let m be the number of MCMC iterations and $\theta^{(i)}$ be the parameter values at the i^{th} iteration ($i = 1, ..., m$). We start by selecting arbitrary initial values for the parameters, $\theta^{(0)}$. Then we propose new parameter values θ^* from a proposal or jumping distribution $h(\theta^*|\theta^{(i-1)})$. We can also select a prior distribution for the parameters, $p(\theta)$, if we have some prior knowledge or intuition of what the values might be.

The Metropolis-Hastings ratio, denoted by r, is the probability of accepting the new proposed parameter values. It is defined as follows:

$$r = \frac{p_{\theta^*}(y_{1:n}) \cdot h(\theta^{(i-1)}|\theta^*) \cdot p(\theta^*)}{p_{\theta^{(i-1)}}(y_{1:n}) \cdot h(\theta^*|\theta^{(i-1)}) \cdot p(\theta^{(i-1)})}.$$

Since r is an acceptance probability, if $r > 1$ we set $r = 1$. The marginal likelihoods $p_{\theta^*}(y_{1:n})$ and $p_{\theta^{(i-1)}}(y_{1:n})$ are estimated using sequential Monte Carlo as follows:

$$\widehat{p}_\theta(y_{1:n}) = \prod_{t=1}^{n} \left(\frac{1}{N} \sum_{k=1}^{N} w_n(X_{1:t}^k) \right). \tag{3.14}$$

At each iteration it must be decided whether or not to accept the proposed parameter values. Let u be a value that is drawn from a Uniform(0,1) distribution. Then the parameters are updated as follows:

- If $u \leq r$, accept θ^* and set $\theta^{(i)} = \theta^*$.
- If $u > r$, reject θ^* and set $\theta^{(i)} = \theta^{(i-1)}$.

The particle Markov chain Monte Carlo algorithm is summarized in Table 3.4. Recall that n is the number of time steps ($t = 1, ..., n$), N is the number of particles ($k = 1, ..., N$) and m is the number of MCMC iterations ($i = 1, ..., m$).

Table 3.4: Particle MCMC Algorithm.

- Select initial parameter values $\theta^{(0)}$.
- Run SMC algorithm with $\theta^{(0)}$ to estimate the marginal likelihood:

$$\widehat{p}_{\theta^{(0)}}(y_{1:n}) = \prod_{t=1}^{n} \left(\frac{1}{N} \sum_{k=1}^{N} w_t(X_{1:t}^k) \right).$$

For $i = 1, ..., m$:

- Propose new parameter values, θ^*.
- Run SMC algorithm with θ^* to estimate the marginal likelihood:

$$\widehat{p}_{\theta^*}(y_{1:n}) = \prod_{t=1}^{n} \left(\frac{1}{N} \sum_{k=1}^{N} w_t(X_{1:t}^k) \right).$$

- Calculate the Metropolis-Hastings Ratio: $r = \dfrac{\hat{p}_{\theta^*}(y_{1:n}) \cdot h(\theta^{(i-1)}|\theta^*) \cdot p(\theta^*)}{\hat{p}_{\theta^{(i-1)}}(y_{1:n}) \cdot h(\theta^*|\theta^{(i-1)}) \cdot p(\theta^{(i-1)})}$.
- Draw u from a Uniform(0,1) distribution. Then update $\theta^{(i)}$ as follows:

$$\text{If } u \leq r, \quad \theta^{(i)} = \theta^*$$
$$\text{If } u > r, \quad \theta^{(i)} = \theta^{(i-1)}.$$

Alternatively, the log of the MH ratio can be used in the PMCMC algorithm. In this case we calculate $\log(r)$ as follows:

$$\log(r) = \log \left[\frac{p_{\theta^*}(y_{1:n})}{p_{\theta^{(i-1)}}(y_{1:n})} \right] + \log \left[\frac{h(\theta^{(i-1)}|\theta^*)}{h(\theta^*|\theta^{(i-1)})} \right] + \log \left[\frac{p(\theta^*)}{p(\theta^{(i-1)})} \right],$$

where $\log \left[\frac{h(\theta^{(i-1)}|\theta^*)}{h(\theta^*|\theta^{(i-1)})} \right]$ is the log proposal ratio and $\log \left[\frac{p(\theta^*)}{p(\theta^{(i-1)})} \right]$ is the log prior ratio. Then we use a similar method to update the parameters, except we comparing $\log(r)$ to $\log(u)$ to decide whether we will accept or reject θ^*. For this project we use the log MH ratio to update the parameters in the PMCMC algorithm.

3.6 Model Comparison

As previously mentioned, there are several different stochastic volatility models that are used in this project. Using the marginal likelihood estimates from the PMCMC algorithm the deviance information criterion (DIC) [28] was calculated for each model in order to compare their effectiveness.

Let $\bar{\theta}$ be the posterior mean or median of $\left\{\theta^{(i)}\right\}_{i=1}^{m}$ and define $D(\bar{\theta})$ as:

$$D(\bar{\theta}) = -2\log\left[p_{\bar{\theta}}(y_{1:n})\right],$$

where the marginal likelihood, $p_{\bar{\theta}}(y_{1:n})$, is approximated using equation (3.14) by running the SMC algorithm with the posterior mean $\bar{\theta}$. Then the DIC is defined as:

$$DIC = D(\bar{\theta}) + 2p_D,$$

where p_D is a penalty term that describes the complexity of the model and penalizes models with more parameters [26]. The penalty term is given by:

$$p_D = \overline{D(\theta)} - D(\bar{\theta}),$$

where $\overline{D(\theta)}$ is approximated by:

$$\overline{D(\theta)} \approx \frac{1}{m}\sum_{i=1}^{m} -2\log\left[p_{\theta^{(i)}}(y_{1:n})\right],$$

where the marginal likelihood, $p_{\theta^{(i)}}(y_{1:n})$, is approximated using equation (3.14) as a byproduct of the SMC algorithm with the parameter $\theta^{(i)}$. The best model will have the smallest DIC.

3.7 Chapter Summary

This chapter introduced the Bayesian approach to estimating stochastic volatility model parameters and posterior distributions. We saw how some of the required integrals cannot be calculated analytically and explored methods of estimating integrals using simulation techniques.

Monte Carlo integration can be used to approximate integrals instead of solving them analytically when it is possible to sample independent draws from the target distribution. Markov chain Monte Carlo allows us to make similar approximations using slightly dependent draws from a Markov chain that has converged to the target distribution.

Importance sampling and sequential importance sampling methods were introduced as alternatives to MCMC for when we cannot sample directly from the target distribution. This lead to the sequential Monte Carlo method by adding a resampling step to the SIS algorithm. However, SMC only provided an estimate for the hidden process $X_{1:n}$, so the particle MCMC method was introduced to estimate the stochastic volatility model parameters.

The particle marginal Metropolis-Hastings (PMMH) method is used to provide an estimate for the stochastic volatility model parameters, θ, and the posterior distribution, $p_\theta(x_{1:n}|y_{1:n})$. The method of proposing and accepting or rejecting new parameter values was outlined and it was shown how the SMC method used to estimate the marginal likelihoods required for calculating the Metropolis-Hastings Ratio. Finally, we introduced the deviance information criterion (DIC) which was used for model comparison.

In the next section simulated data sets will be generated for each of the stochastic volatility models introduced in Chapter 2 and the particle MCMC method will be used to estimate the model parameters.

Chapter 4

Simulation Studies

4.1 Gibbs Sampler and PMCMC for Basic SVM Version 1

4.1.1 PMCMC

The basic stochastic volatility model from Section 2.2.1 was used to create the simulated data set shown in Figure 4.3. To generate this data set the model parameters were set to $\alpha = 0.9$, $\beta^2 = 1.42$ and $\sigma^2 = 0.58$ for $n = 500$ time steps. In this section both the Gibbs sampler and PMCMC methods were used to estimate the model parameters based on the simulated observations, $Y_{1:n}$.

The PMCMC algorithm was run using $N = 10000$ particles and $m = 10000$ MCMC iterations. The initial parameter values were set to $\alpha = 0.92$, $\beta^2 = 1$ and $\sigma^2 = 0.5$. As previously mentioned the Markov chain should converge regardless of the starting values, so the starting point will only affect the time required for the chain to converge. We assumed the following prior distributions for the model parameters: $\alpha \sim Uniform(-1, 1)$, $\beta^2 \sim IG(0.01, 0.01)$ and $\sigma^2 \sim IG(0.01, 0.01)$. The parameters were updated using a random walk with the following proposal distributions:

$$\alpha^* \sim TN\left(\alpha^{(i-1)}, \tau_\alpha^2, -1, 1\right),$$

$$\log\left(\beta^{2*}\right) \sim N\left(\log\left(\beta^{2\,(i-1)}\right), \tau_{\beta^2}^2\right),$$

$$\log\left(\sigma^{2*}\right) \sim N\left(\log\left(\sigma^{2\,(i-1)}\right), \tau_{\sigma^2}^2\right),$$

where α^* follows a truncated normal distribution over the range $(-1, 1)$ and β^2 and σ^2 follow a log normal distribution. The scale parameters were set to $\tau_\alpha = 0.01$, $\tau_{\beta^2} = 0.3$ and $\tau_{\sigma^2} = 0.3$.

Figure 4.1 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm. A burn-in of 2500 iterations was used to ensure that the Markov chain had converged. The histograms of the posterior draws, after the burn-in period, are displayed in Figure 4.2. The acceptance rate for the proposed parameters was 0.2514.

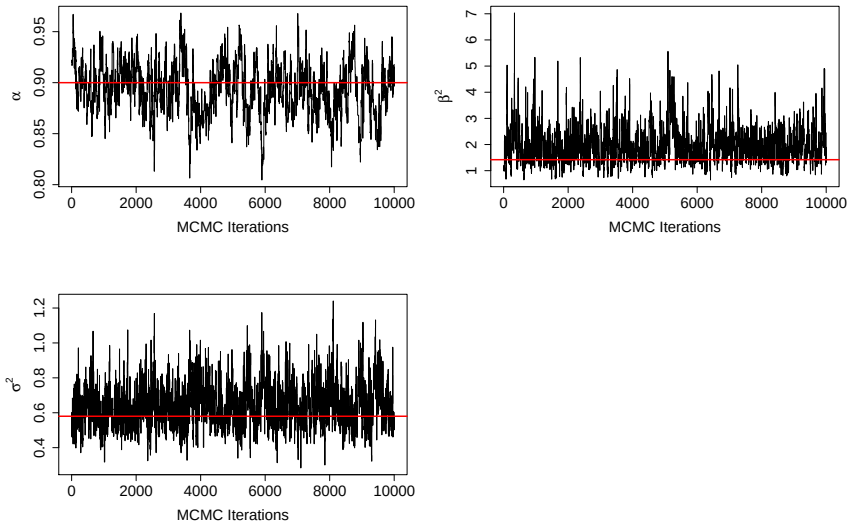

Figure 4.1: Basic SVM Version 1 trace plots of α, μ_x, μ_y and σ^2 resulting from PMCMC for simulated data.

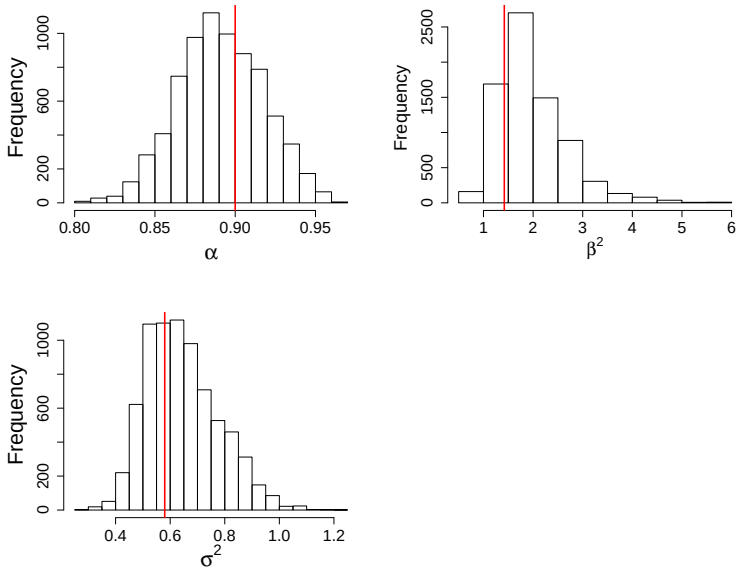

Figure 4.2: Basic SVM Version 1 histograms of α, μ_x, μ_y and σ^2 resulting from PMCMC for simulated data.

The posterior means of the parameters from PMCMC were used to generate the estimated X values shown in Figure 4.3.

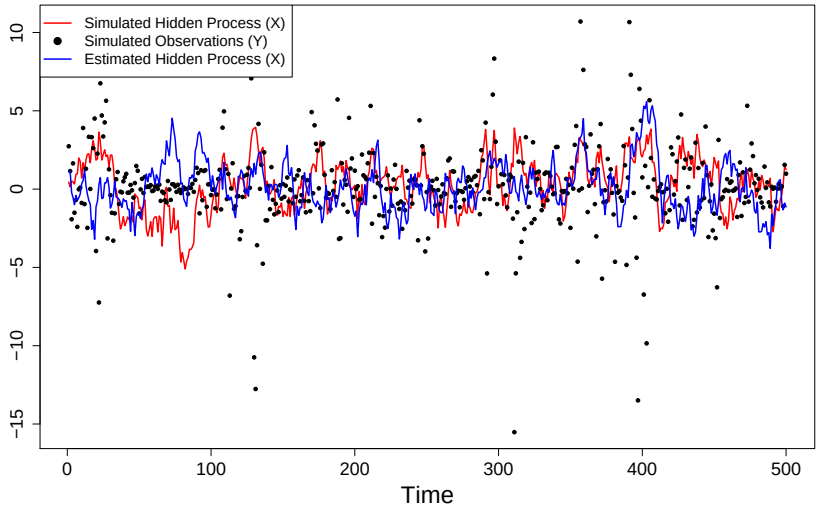

Figure 4.3: Basic SVM Version 1 simulated data and PMCMC estimates, with observations $Y_{1:n}$ and hidden process $X_{1:n}$ for simulated data.

4.1.2 Gibbs Sampler

Similarly, the Gibbs algorithm, outlined in Section 3.3.2, was run using $m = 10000$ MCMC iterations and the same initial parameter values. Figure 4.4 shows the trace plots of the parameter posterior draws from each iteration of the Gibbs Sampler algorithm. The histograms of the posterior draws, after a burn-in of 2500 iterations, are displayed in Figure 4.5.

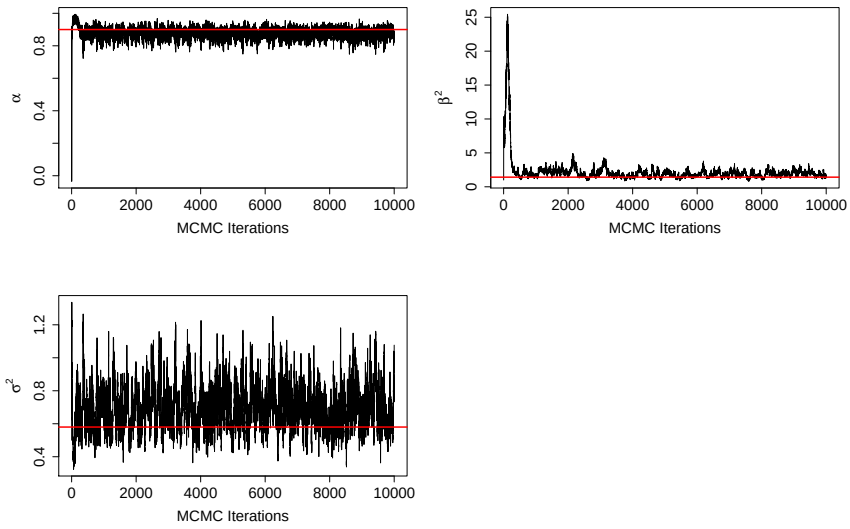

Figure 4.4: Basic SVM Version 1 trace plots of α, μ_x, μ_y and σ^2 resulting from Gibbs Sampler for simulated data.

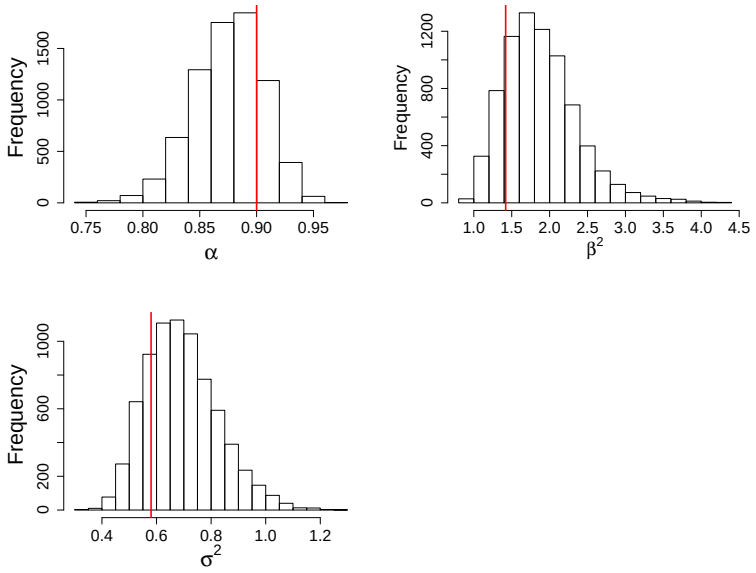

Figure 4.5: Basic SVM Version 1 histograms of α, μ_x, μ_y and σ^2 resulting from Gibbs Sampler for simulated data.

4.1.3 Comparison of PMCMC and Gibbs Sampler

Both PMCMC and Gibbs Sampler were used to estimate the model parameters based on the simulated observations.The true parameter values and a summary of the parameter posterior distributions are given in Table 4.1.

Table 4.1: Summary of Basic SVM Version 1 parameter posterior distributions resulting from PMCMC and Gibbs Sampler for simulated data.

		Gibbs Sampler		PMCMC	
Parameters	True Value	Mean	95% CI	Mean	95% CI
α	0.9	0.875	(0.823, 0.923)	0.891	(0.847, 0.935)
β^2	1.42	1.875	(1.208, 2.731)	1.98	(1.134, 3.248)
σ^2	0.58	0.694	(0.502, 0.933)	0.644	(0.458, 0.882)

4.2 PMCMC for Basic SVM Version 2

The basic stochastic volatility model from Section 2.2.2 was used to create the simulated data set shown in Figure 4.8. To generate this data set the model parameters were set to $\alpha = 0.88$, $\mu_x = 3.5$, $\mu_y = 0.35$ and $\sigma^2 = 0.58$ for $n = 500$ time steps.

In this section the PMCMC method was used to estimate the model parameters based on the simulated observations, $Y_{1:n}$. The PMCMC algorithm was run using $N = 10000$ particles and $m = 10000$ MCMC iterations. The initial parameter values were set to $\alpha = 0.9$, $\mu_x = 0$, $\mu_y = 0$ and $\sigma^2 = 0.5$. We assumed the following prior distributions for the model parameters: $\alpha \sim TN(0.9, 0.1, -1, 1)$, $\mu_x \sim N(0, 10)$, $\mu_y \sim N(0, 10)$ and $\sigma^2 \sim IG(1, 1)$. The parameters were updated using a random walk with the following proposal distributions:

$$\alpha^* \sim TN\left(\alpha^{(i-1)}, \tau_\alpha^2, -1, 1\right),$$

$$\mu_x^* \sim N\left(\mu_x^{(i-1)}, \tau_{\mu_x}^2\right),$$

$$\mu_y^* \sim N\left(\mu_y^{(i-1)}, \tau_{\mu_y}^2\right),$$

$$\log\left(\sigma^{2*}\right) \sim N\left(\log\left(\sigma^{2\,(i-1)}\right), \tau_{\sigma^2}^2\right).$$

The scale parameters were set to $\tau_\alpha = 0.1$, $\tau_{\mu_x} = 0.32$, $\tau_{\mu_y} = 0.03$ and $\tau_{\sigma^2} = 0.1$.

Figure 4.6 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm. A burn-in of 2500 iterations was used to ensure that the Markov chain had converged. The histograms of the posterior draws, after the burn-in period, are displayed in Figure 4.7.

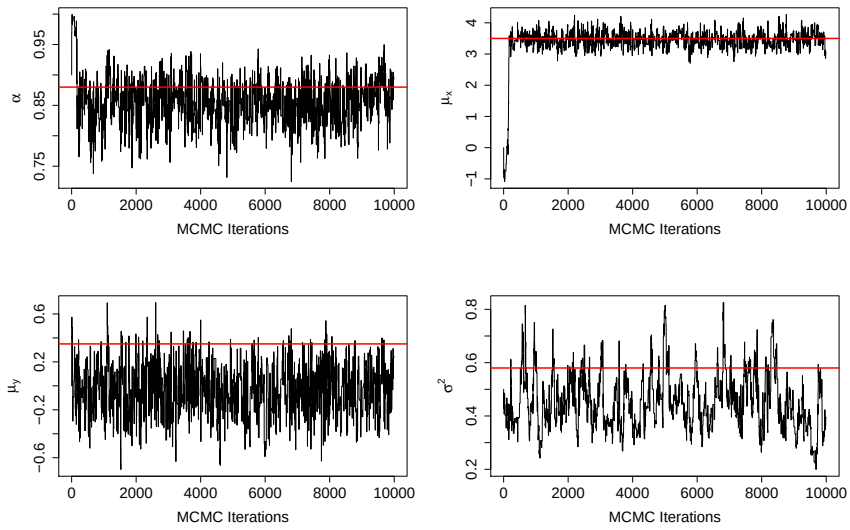

Figure 4.6: Basic SVM Version 2 trace plots of α, μ_x, μ_y and σ^2 resulting from PMCMC for simulated data.

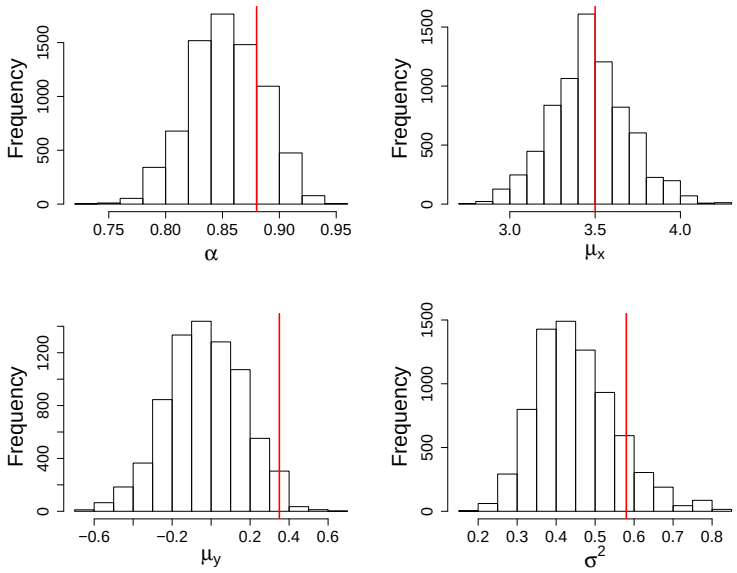

Figure 4.7: Basic SVM Version 2 histograms of α, μ_x, μ_y and σ^2 resulting from PMCMC for simulated data.

The acceptance rate for the proposed parameters was 0.1598. Table 4.2 gives a summary of the parameter posterior distributions resulting from the PMCMC algorithm.

Table 4.2: Summary of Basic SVM Version 2 parameter posterior distributions resulting from PMCMC for simulated data.

Parameter	True Value	Mean	95% CI
α	0.88	0.853	(0.799, 0.905)
μ_x	3.5	3.467	(3.097, 3.863)
μ_y	0.35	-0.032	(-0.374, 0.294)
σ^2	0.58	0.452	(0.305, 0.642)

The posterior means of the parameters were used to generate the estimated X values shown in Figure 4.8.

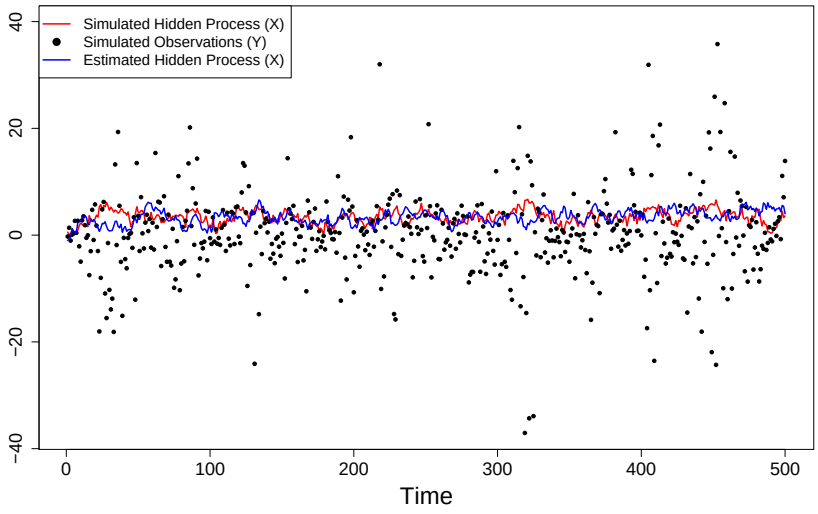

Figure 4.8: Basic SVM Version 2 simulated data and PMCMC estimates, with observations $Y_{1:n}$ and hidden process $X_{1:n}$ for simulated data.

4.3 PMCMC for SVM with Fat Tails

The stochastic volatility model with fat tails from Section 2.3 was used to create the simulated data set shown in Figure 4.11. To generate this data set the model parameters were set to $\alpha = 0.91$, $\mu_x = 3.4$, $df = 10$ and $\sigma^2 = 0.32$ for $n = 500$ time steps.

The PMCMC method was used to estimate the model parameters based on the simulated observations, $Y_{1:n}$. The PMCMC algorithm was run using $N = 10000$ particles and $m = 10000$ MCMC iterations. The initial parameter values were set to $\alpha = 0.9$, $\mu_x = 0$, $df = 10$ and $\sigma^2 = 0.4$. We assumed the following prior distributions for the model parame-

ters: $\alpha \sim TN(0.9, 10, -1, 1)$, $\mu_x \sim N(0, 10)$, $df \sim TN(20, 10, 2, \infty)$ and $\sigma^2 \sim IG(1, 1)$. The parameters were updated using a random walk with the following proposal distributions:

$$\alpha^* \sim TN\left(\alpha^{(i-1)}, \tau_\alpha^2, -1, 1\right),$$

$$\mu_x^* \sim N\left(\mu_x^{(i-1)}, \tau_{\mu_x}^2\right),$$

$$df^* \sim TN\left(df^{(i-1)}, \tau_{df}^2, 2, \infty\right),$$

$$\log\left(\sigma^{2*}\right) \sim N\left(\log\left(\sigma^{2\,(i-1)}\right), \tau_{\sigma^2}^2\right).$$

The scale parameters were set to $\tau_\alpha = 0.05$, $\tau_{\mu_x} = 0.32$, $\tau_{df} = 3$ and $\tau_{\sigma^2} = 0.1$.

Figure 4.9 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm. The histograms of the posterior draws, after a burn-in period of 2500 iterations, are displayed in Figure 4.10.

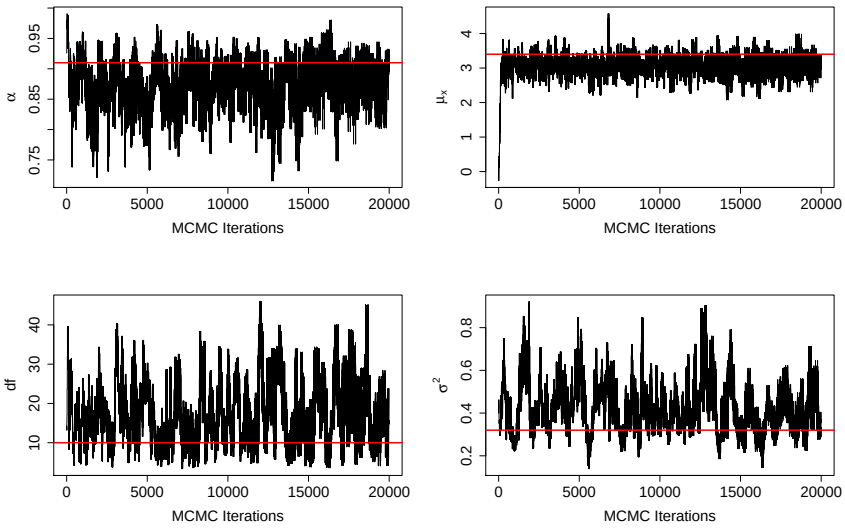

Figure 4.9: SVM Fat Tails trace plots of α, μ_x, df and σ^2 resulting from PMCMC for simulated data.

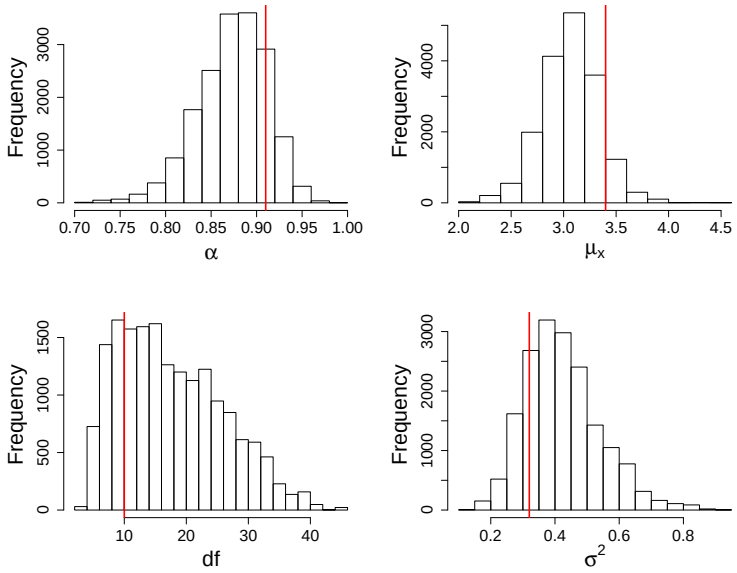

Figure 4.10: SVM Fat Tails histograms of α, μ_x, df and σ^2 resulting from PMCMC for simulated data.

The acceptance rate for the proposed parameters was 0.3514. Table 4.3 gives a summary of the parameter posterior distributions resulting from the PMCMC algorithm.

Table 4.3: Summary of SVM Fat Tails parameter posterior distributions resulting from PMCMC for simulated data.

Parameter	True Value	Mean	95% CI
α	0.91	0.873	(0.805, 0.929)
μ_x	3.4	3.063	(2.609, 3.498)
df	10	17.54	(6.157, 32.635)
σ^2	0.32	0.423	(0.263, 0.637)

The posterior means of the parameters were used to generate the estimated X values shown in Figure 4.11.

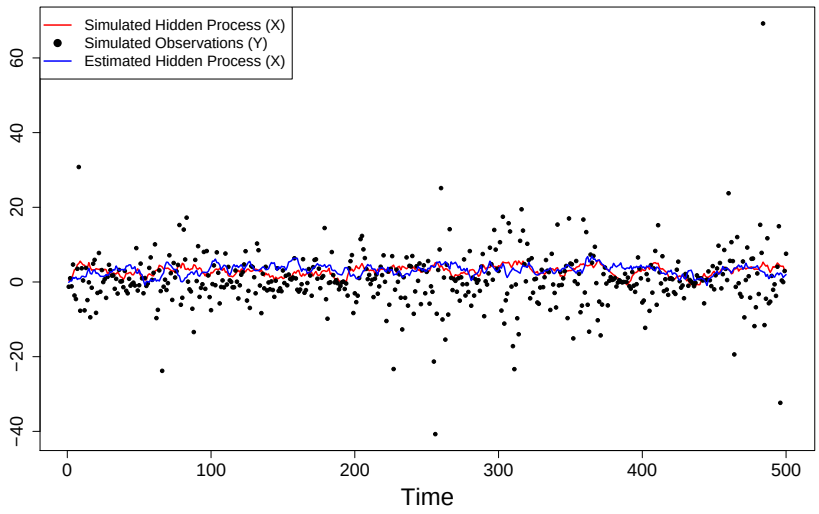

Figure 4.11: SVM Fat Tails simulated data and PMCMC estimates, with observations $Y_{1:n}$ and hidden process $X_{1:n}$ for simulated data.

4.4 PMCMC for SVM with Leverage Effect

The stochastic volatility model with leverage effect from Section 2.4 was used to create the simulated data set shown in Figure 4.14. To generate this data set the model parameters were set to $\alpha = 0.89$, $\mu_x = 3.6$, $\rho = -0.05$ and $\sigma^2 = 0.47$ for $n = 500$ time steps.

The PMCMC method was used to estimate the model parameters based on the simulated observations, $Y_{1:n}$. The PMCMC algorithm was run using $N = 10000$ particles and $m = 10000$ MCMC iterations. The initial parameter values were set to $\alpha = 0.9$, $\mu_x = 0$, $\rho = 0$ and $\sigma^2 = 0.4$. We assumed the following prior distributions for the model parameters: $\alpha \sim TN(0.9, 10, -1, 1)$, $\mu_x \sim N(0, 10)$, $\rho \sim TN(0.5, 10, -1, 1)$ and $\sigma^2 \sim IG(0.01, 0.01)$. The parameters were updated using a random walk with the following proposal distributions:

$$\alpha^* \sim TN\left(\alpha^{(i-1)}, \tau_\alpha^2, -1, 1\right),$$

$$\mu_x^* \sim N\left(\mu_x^{(i-1)}, \tau_{\mu_x}^2\right),$$

$$\rho^* \sim TN\left(\rho^{(i-1)}, \tau_\rho^2, -1, 1\right),$$

$$\log\left(\sigma^{2*}\right) \sim N\left(\log\left(\sigma^{2\,(i-1)}\right), \tau_{\sigma^2}^2\right).$$

The scale parameters were set to $\tau_\alpha = 0.03$, $\tau_{\mu_x} = 0.03$, $\tau_\rho = 0.03$ and $\tau_{\sigma^2} = 0.03$.

Figure 4.12 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm. The histograms of the posterior draws, after a burn-in period of 5000 iterations, are displayed in Figure 4.13.

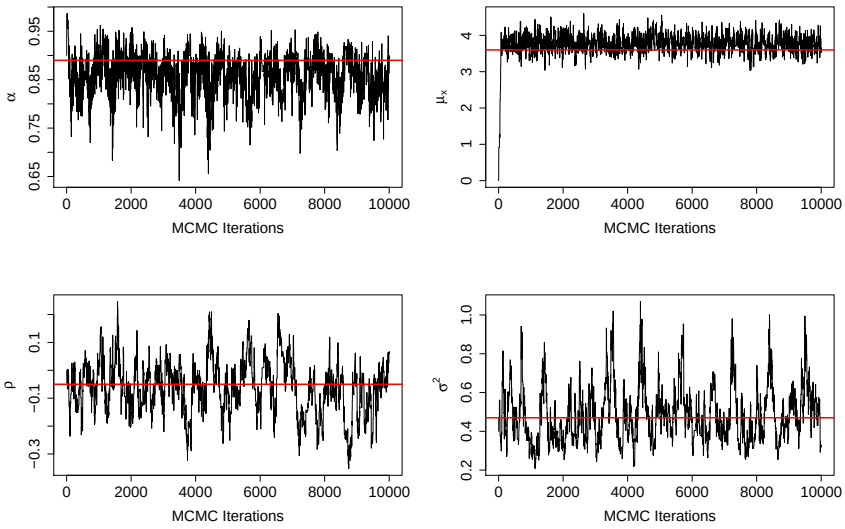

Figure 4.12: SVM with Leverage Effect trace plots of α, μ_x, ρ and σ^2 resulting from PMCMC for simulated data.

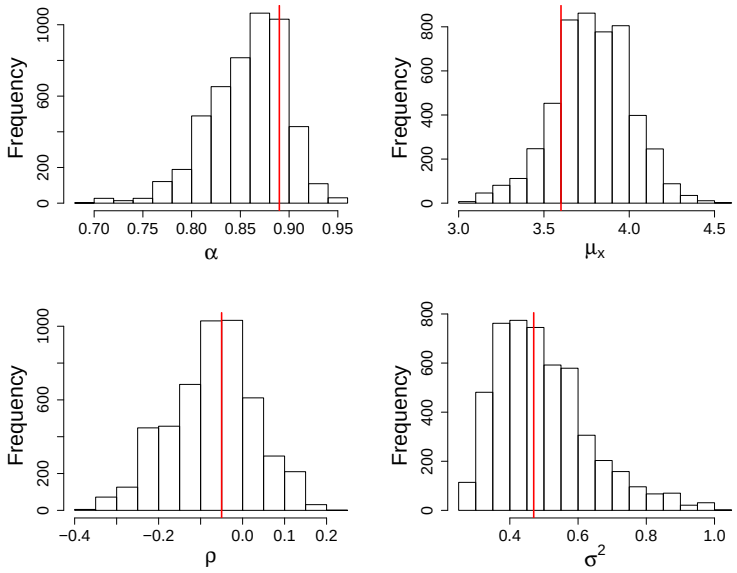

Figure 4.13: SVM with Leverage Effect histograms of α, μ_x, ρ and σ^2 resulting from PMCMC for simulated data.

The acceptance rate for the proposed parameters was 0.2441. Table 4.4 gives a summary of the parameter posterior distributions resulting from the PMCMC algorithm.

Table 4.4: Summary of SVM with Leverage Effect parameter posterior distributions resulting from PMCMC for simulated data.

Parameter	True Value	Mean	95% CI
α	0.89	0.857	(0.788, 0.914)
μ_x	3.6	3.783	(3.405, 4.136)
ρ	-0.05	-0.073	(-0.248, 0.095)
σ^2	0.47	0.497	(0.313, 0.766)

The posterior means of the parameters were used to generate the estimated X values shown in Figure 4.14.

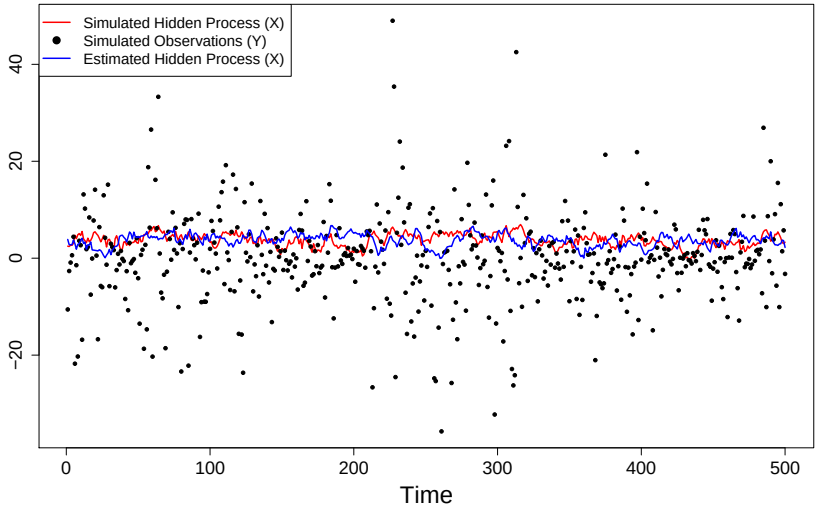

Figure 4.14: SVM with Leverage Effect simulated data and PMCMC estimates, with observations $Y_{1:n}$ and hidden process $X_{1:n}$ for simulated data.

4.5 PMCMC for SVM with Covariate Effects

The stochastic volatility model with covariate effects from Section 2.5 was used to create the simulated data set shown in Figure 4.17. To generate this data set the model parameters were set to $\alpha = 0.88$, $\mu_x = 3.6$, $\eta = (0.88, -0.25)$ and $\sigma^2 = 0.55$ for $n = 500$ time steps.

The two covariates used for this simulation are Bitcoin data sets that will be introduced in the next chapter.

In this section the PMCMC method was used to estimate the model parameters based on the simulated observations, $Y_{1:n}$. The PMCMC algorithm was run using $N = 10000$ particles and $m = 10000$ MCMC iterations. The initial parameter values were set to $\alpha = 0.9$, $\mu_x = 0.2$, $\eta = (0, 0)$ and $\sigma^2 = 0.35$. We assumed the following prior distributions for the model parameters: $\alpha \sim TN(0.9, 10, -1, 1)$, $\mu_x \sim N(0, 10)$, $\eta \sim N(0, 10)$ and $\sigma^2 \sim IG(0.01, 0.01)$. The parameters were updated using a random walk with the following proposal distributions:

$$\alpha^* \sim TN\left(\alpha^{(i-1)}, \tau_\alpha^2, -1, 1\right),$$

$$\mu_x^* \sim N\left(\mu_x^{(i-1)}, \tau_{\mu_x}^2\right),$$

$$\eta^* \sim N\left(\eta^{(i-1)}, \tau_\eta^2\right),$$

$$\log\left(\sigma^{2*}\right) \sim N\left(\log\left(\sigma^{2\,(i-1)}\right), \tau_{\sigma^2}^2\right).$$

The scale parameters were set to $\tau_\alpha = 0.05$, $\tau_{\mu_x} = 0.3162$, $\tau_\eta = 0.2$ and $\tau_{\sigma^2} = 0.1$.

Figure 4.15 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm. The histograms of the posterior draws, after a burn-in period of 5000 iterations, are displayed in Figure 4.16.

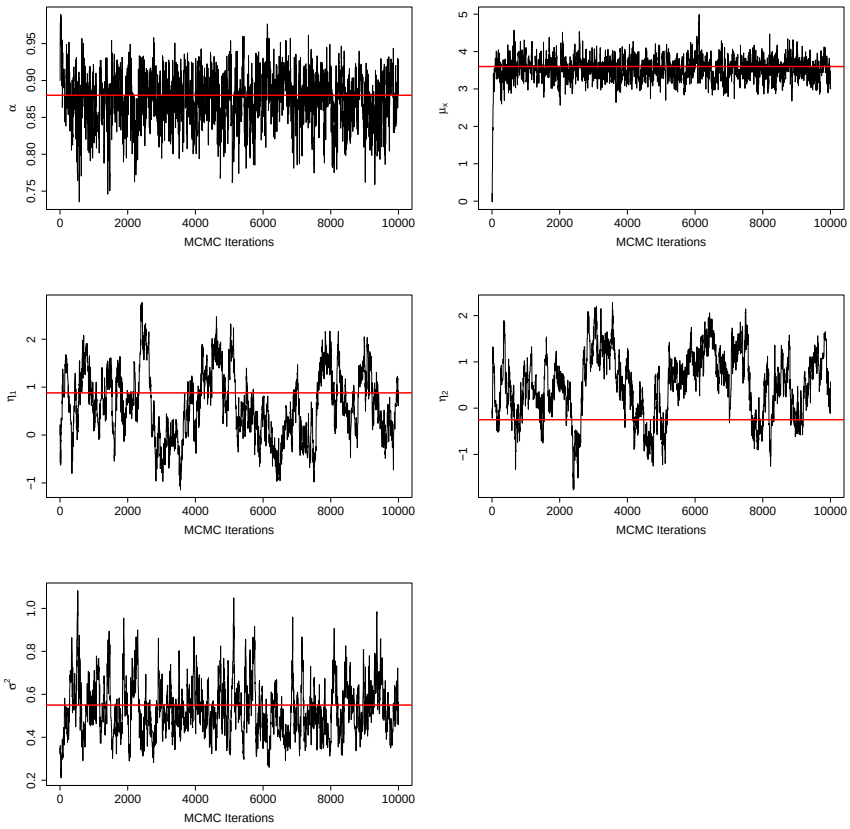

Figure 4.15: SVM with Covariate Effect trace plots of α, μ_x, η_1, η_2 and σ^2 resulting from PMCMC for simulated data.

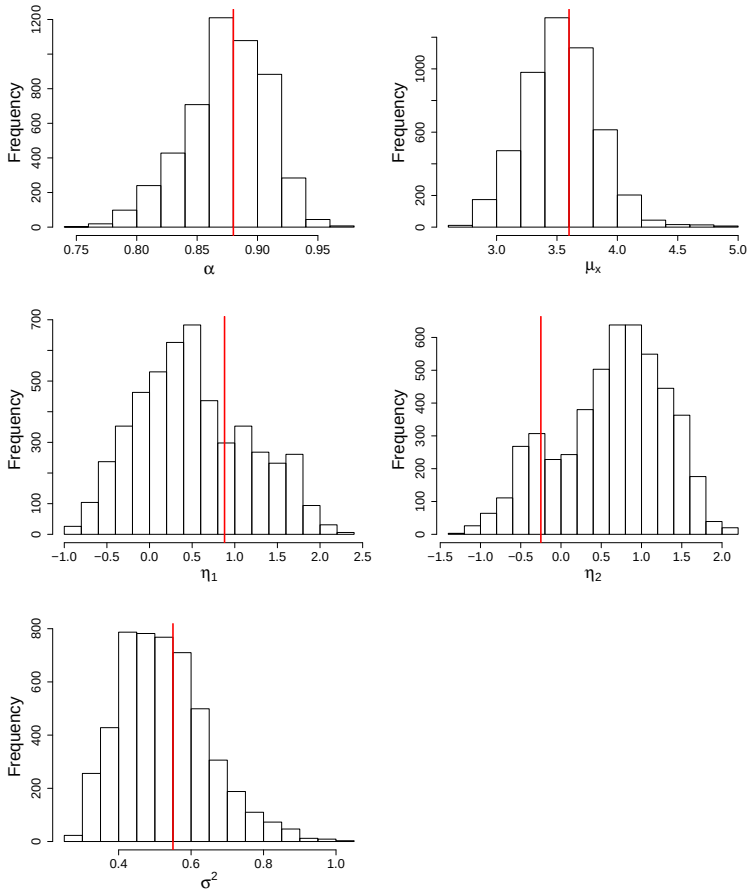

Figure 4.16: SVM with Covariate Effect histograms of α, μ_x, η_1, η_2 and σ^2 resulting from PMCMC for simulated data.

The acceptance rate for the proposed parameters was 0.2925. Table 4.5 gives a summary of the parameter posterior distributions resulting from the PMCMC algorithm.

Table 4.5: Summary of SVM with Covariate Effect parameter posterior distributions resulting from PMCMC for simulated data.

Parameter	True Value	Mean	95% CI
α	0.88	0.874	(0.814, 0.924)
μ_x	3.6	3.539	(3.041, 4.008)
η_1	0.88	0.516	(-0.491, 1.694)
η_2	-0.25	0.627	(-0.549, 1.59)
σ^2	0.55	0.526	(0.346, 0.75)

The posterior means of the parameters were used to generate the estimated X values shown in Figure 4.17.

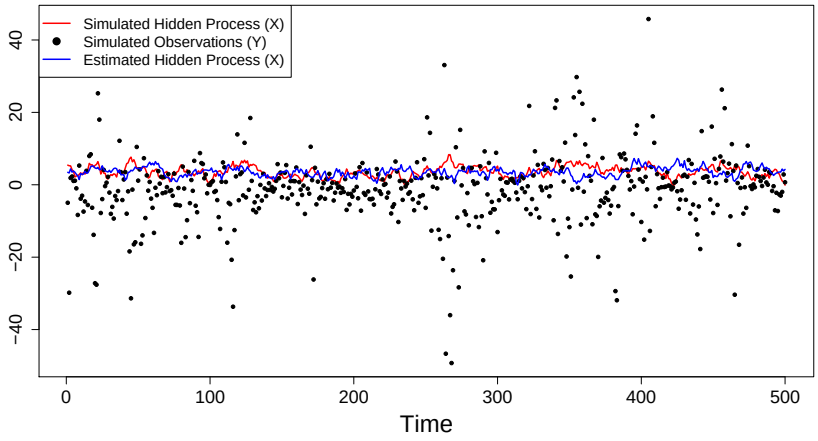

Figure 4.17: SVM with Covariate Effect simulated data and PMCMC estimates, with observations $Y_{1:n}$ and hidden process $X_{1:n}$ for simulated data.

4.6 Chapter Summary

In this chapter simulation studies were conducted to demonstrate the capabilities of the particle Markov chain Monte Carlo algorithm. Simulated data sets were generated using the stochastic volatility models introduced in Chapter 2. Particle MCMC was used to estimate the model parameters based on the simulated observations and compared to the true parameter values used to generate the data.

The PMCMC algorithm was found to be very sensitive to the choice of scale parameters, τ, for the proposal distributions. If a scale parameter is too large this can lead to a low acceptance rate and make the algorithm inefficient. This is because the steps in the random walk are too large and too many candidate draws are being rejected. On the other hand, a scale parameter that is too small can lead to an acceptance rate that is too high because the steps in the random walk are too small and the chain is not moving around the parameter space quickly enough. Choosing the optimal scale parameters for the proposal distributions is a difficult task, especially when dealing with a model that has multiple parameters such as the stochastic volatility models used in this study.

In the next chapter stochastic volatility models are used to model real Bitcoin exchange rate data and the model parameters are estimated using particle MCMC.

Chapter 5

Applications to Bitcoin Exchange Rate Data

5.1 Data

The data set contains daily Bitcoin exchange rates (Bitcoin versus USD) over the period of June 30, 2014 to June 30, 2016. The daily values are a weighted average of the exchange rates from the largest Bitcoin exchanges. The data was obtained from the Quandl data base, https://www.quandl.com/collections/markets/bitcoin-data. Although data is available for more than just the past two years, the Bitcoin exchange rate was extremely volatile before this period and likely followed very different patterns than it does today.

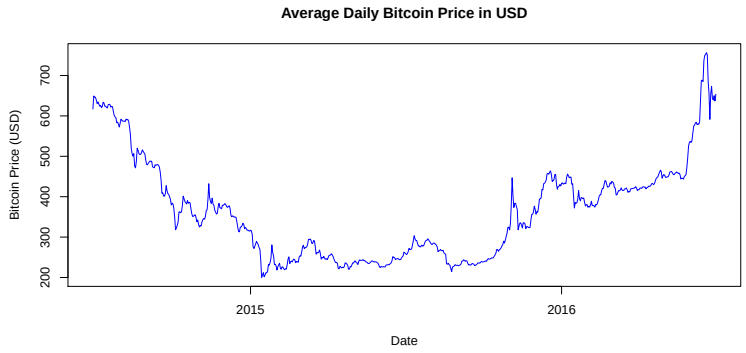

Figure 5.1: Daily Bitcoin exchange rate.

The particle MCMC algorithm was applied to the relative change in the exchange rate of Bitcoin. Let $x(t)$ be a time series at time t, then the relative change, $r(t)$, is defined as:

$$r(t) = \frac{x(t)}{x(t-1)}.$$

The relative change in the daily Bitcoin exchange rate data is shown in Figure 5.2.

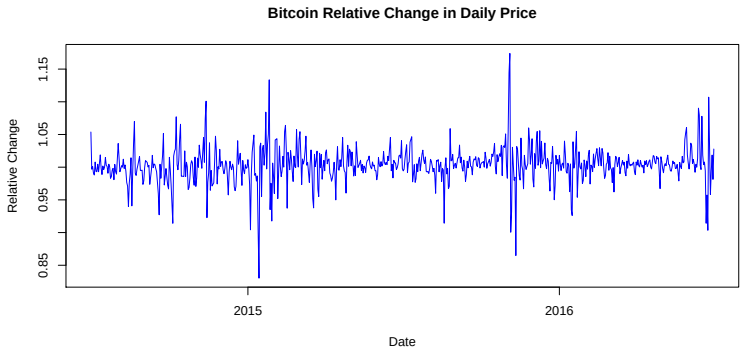

Figure 5.2: Relative change in daily Bitcoin exchange rate.

In addition to price and exchange rates, the Quandl Bitcoin database has many other data sets from various aspects of the Bitcoin network, such as market size or network activity. Some of these variables could possibly be related to Bitcoin price and could be used in a stochastic volatility model with covariate effects. Specifically, we will consider two covariates: the number of Bitcoin transactions per day and the number of unique Bitcoin addresses used per day. These data sets are shown in Figures 5.3 and 5.4, respectively.

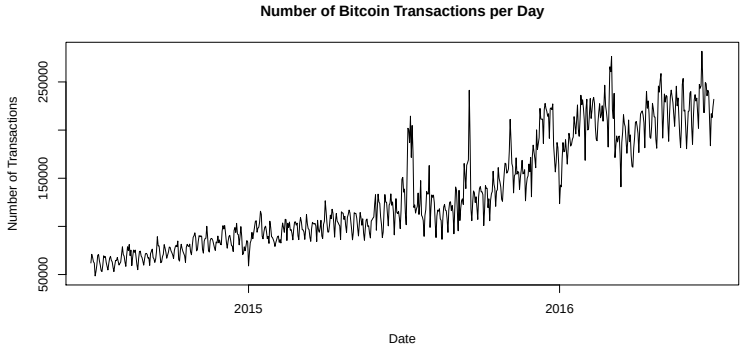

Figure 5.3: Number of Bitcoin Transactions per Day.

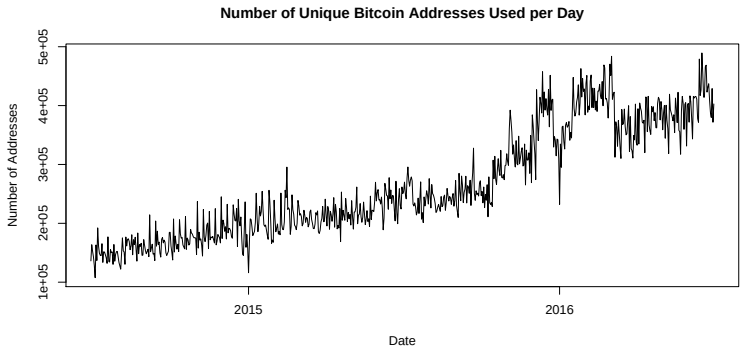

Figure 5.4: Number of Unique Bitcoin Addresses Used per Day.

In the following sections the stochastic volatility models introduced in Chapter 2 are used to model the Bitcoin exchange rate data and particle Markov chain Monte Carlo was used to estimate model parameters. The Bitcoin data analysis begins with the basic stochastic volatility model. Only version 2 of the basic SVM was applied to the Bitcoin data since both versions are just different parameterizations of the same model.

5.2 Bitcoin Data Analysis with Basic SVM

The PMCMC algorithm was run using $N = 15000$ particles and $m = 41300$ MCMC iterations. The initial parameters were set to $\alpha_0 = 0.9$, $\mu_{x_0} = 0$, $\mu_{y_0} = 0.2$ and $\sigma_0^2 = 0.4$. The parameters are assumed to have the same prior distribution as in Section 4.2 and are updated using the same random walk method with proposal distributions of the same form. For this application the scale parameters were set to $\tau_\alpha = 0.1$, $\tau_{\mu_x} = 0.3162$, $\tau_{\mu_y} = 0.03$ and $\tau_{\sigma^2} = 0.1$.

Figure 5.5 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm. A burn-in of 2500 iterations was used to ensure that the Markov chain had converged. The histograms of the posterior draws, after the burn-in period, are displayed in Figure 5.6.

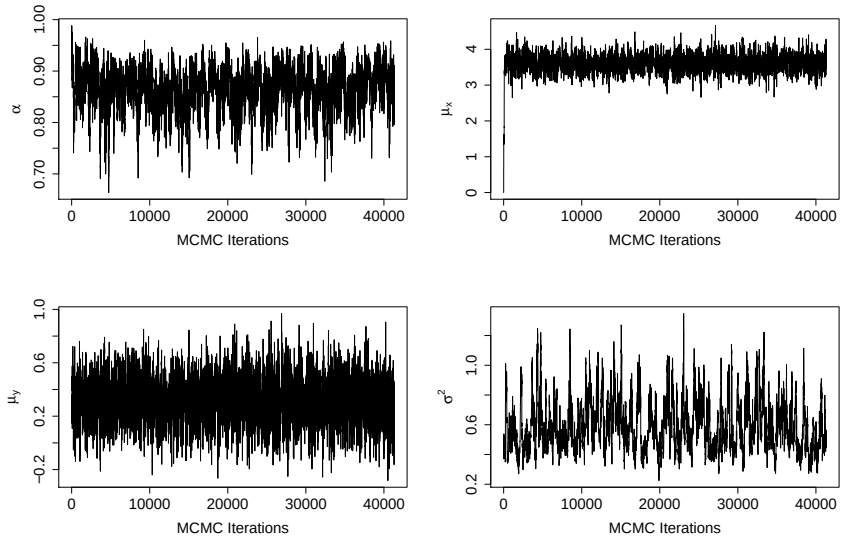

Figure 5.5: Basic SVM Version 2 trace plots of α, μ_x, μ_y and σ^2 resulting from PMCMC for Bitcoin data.

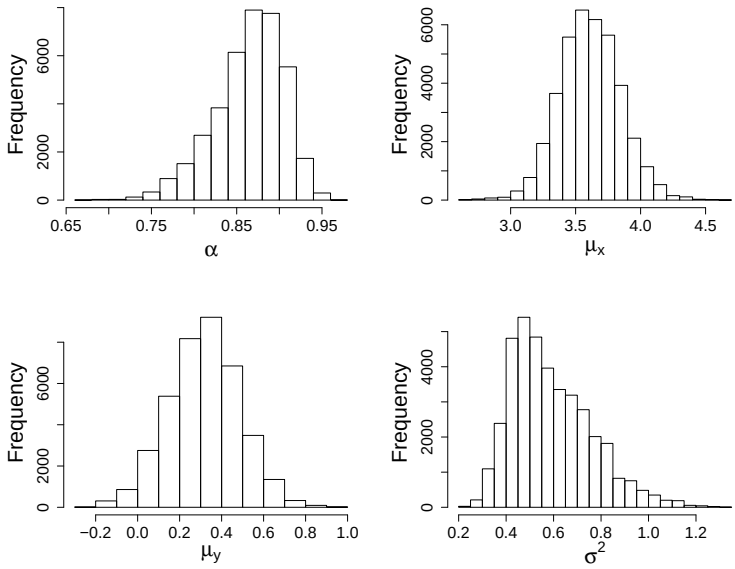

Figure 5.6: Basic SVM Version 2 histograms of α, μ_x, μ_y and σ^2 resulting from PMCMC for Bitcoin data.

Table 5.1 gives the posterior mean and 95% credible interval resulting from the PMCMC algorithm for each of the model parameters. The acceptance rate was 0.1185 and the $DIC = 4910.634$.

Table 5.1: Summary of Basic SVM Version 2 parameter posterior distributions resulting from PMCMC for Bitcoin data.

Parameter	Mean	95% CI
α	0.868	(0.785, 0.917)
μ_x	3.621	(3.236, 3.984)
μ_y	0.334	(0.062, 0.609)
σ^2	0.571	(0.38, 0.938)

5.3 Bitcoin Data Analysis for SVM with Fat Tail

The PMCMC algorithm was run using $N = 5000$ particles and $m = 10000$ MCMC iterations. The initial parameters were set to $\alpha_0 = 0.9$, $\mu_{x_0} = 0$, $df_0 = 10$ and $\sigma_0^2 = 0.4$. The parameters are assumed to have the same prior distribution as in Section 4.3 and are updated using the same random walk method with proposal distributions of the same form. For this application the scale parameters were set to $\tau_\alpha = 0.05$, $\tau_{\mu_x} = 0.3162$, $\tau_{df} = 3$ and $\tau_{\sigma^2} = 0.1$.

Figure 5.7 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm.The histograms of the posterior draws, after a burn-in period of 2500 iterations, are displayed in Figure 5.8.

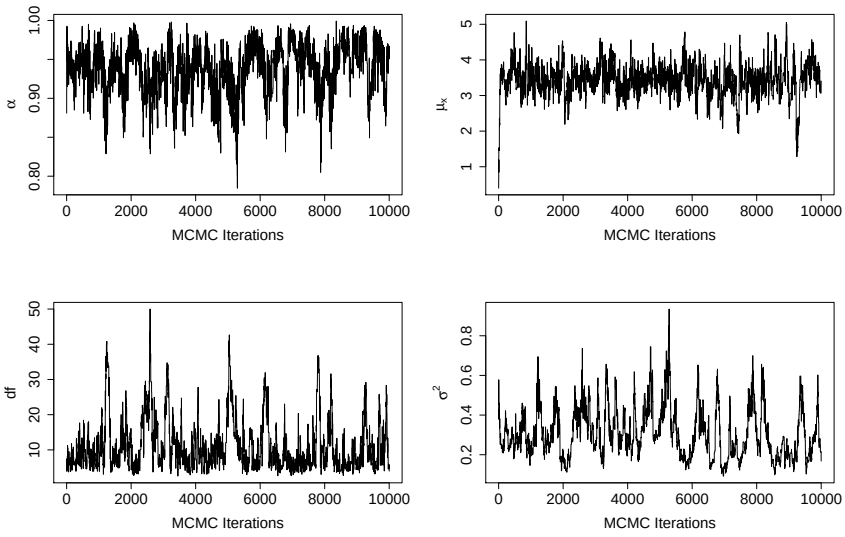

Figure 5.7: SVM with Fat Tails trace plots of α, μ_x, df and σ^2 resulting from PMCMC for Bitcoin data.

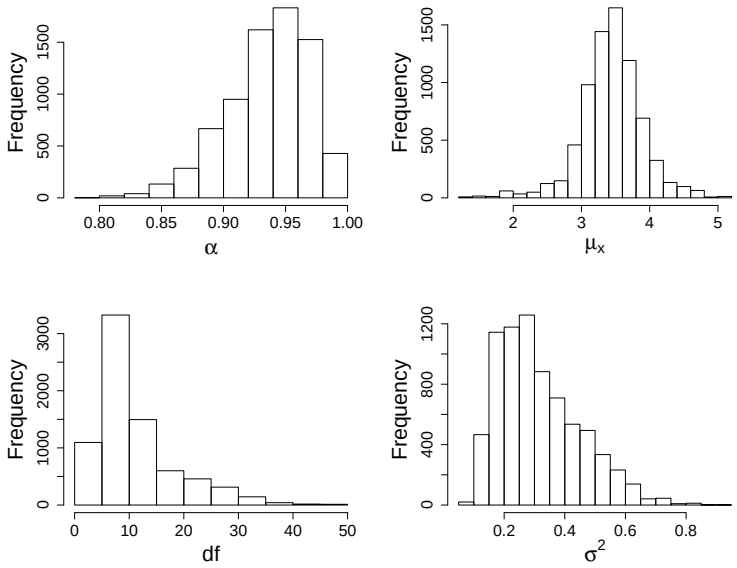

Figure 5.8: SVM with Fat Tails histograms of α, μ_x, df and σ^2 resulting from PMCMC for Bitcoin data.

Table 5.2 gives the posterior mean and 95% credible interval resulting from the PMCMC algorithm for each of the model parameters. The acceptance rate was 0.2648 and the $DIC = 4914.64$.

Table 5.2: Summary of SVM with Fat Tails parameter posterior distributions resulting from PMCMC for Bitcoin data.

Parameter	Mean	95% CI
α	0.936	(0.875, 0.982)
μ_x	3.439	(2.713, 4.142)
df	11.11	(4.099, 27.172)
σ^2	0.3144	(0.146, 0.578)

5.4 Bitcoin Data Analysis for SVM with Leverage Effect

The PMCMC algorithm was run using $N = 15000$ particles and $m = 33400$ MCMC iterations. The initial parameters were set to $\alpha_0 = 0.9$, $\mu_{x_0} = 0$, $\rho_0 = 0$ and $\sigma_0^2 = 0.4$. The parameters are assumed to have the same prior distribution as in Section 4.4 and are updated using the same random walk method with proposal distributions of the same form.

For this application the scale parameters were set to $\tau_\alpha = 0.1$, $\tau_{\mu_x} = 0.3162$, $\tau_\rho = 0.035$ and $\tau_{\sigma^2} = 0.1$.

Figure 5.9 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm.The histograms of the posterior draws, after a burn-in period of 5000 iterations, are displayed in Figure 5.10.

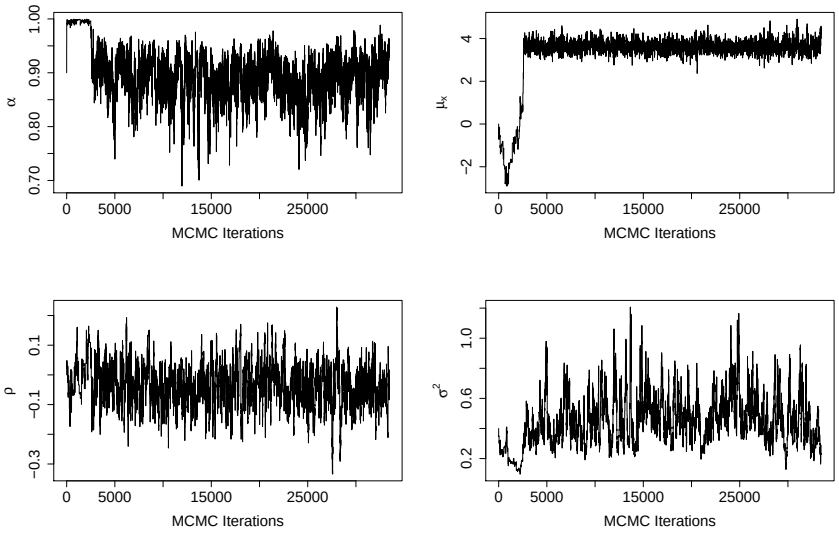

Figure 5.9: SVM with Leverage Effect trace plots of α, μ_x, ρ and σ^2 resulting from PMCMC for Bitcoin data.

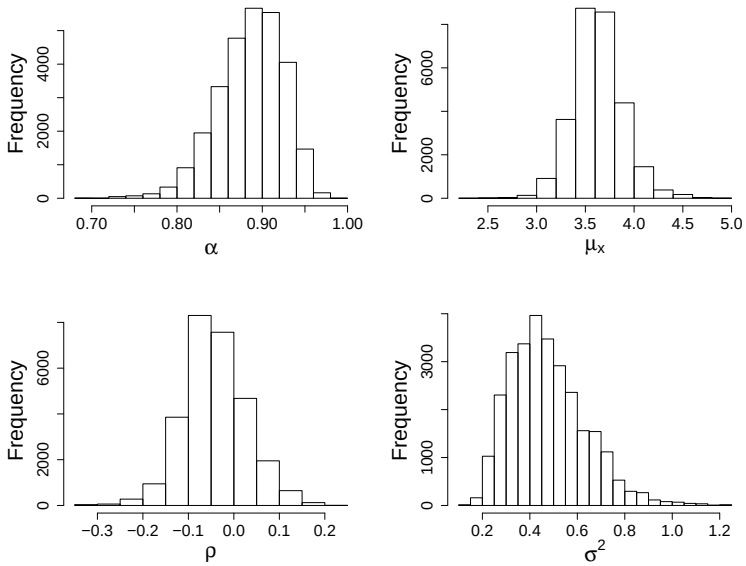

Figure 5.10: SVM with Leverage Effect histograms of α, μ_x, ρ and σ^2 resulting from PMCMC for Bitcoin data.

Table 5.3 gives the posterior mean and 95% credible interval resulting from the PMCMC algorithm for each of the model parameters. The acceptance rate was 0.173 and the $DIC = 4903.698$.

Table 5.3: Summary of SVM with Leverage Effect parameter posterior distributions resulting from PMCMC for Bitcoin data.

Parameter	Mean	95% CI
α	0.886	(0.819, 0.942)
μ_x	3.628	(3.231, 4.05)
μ_y	-0.041	(-0.147, 0.078)
σ^2	0.475	(0.256, 0.754)

5.5 Bitcoin Data Analysis for SVM with Covariate Effect

The PMCMC algorithm was run using $N = 5000$ particles and $m = 30000$ MCMC iterations. The initial parameters were set to The initial parameters were set to $\alpha_0 = 0.9$, $\mu_{x_0} = 0.2$, $\eta_0 = (0, 0)$ and $\sigma_0^2 = 0.35$. The parameters are assumed to have the same prior distribution as in Section 4.5 and are updated using the same random walk method with proposal distributions of the same form. For this application the scale parameters were set to $\tau_\alpha = 0.05$, $\tau_{\mu_x} = 0.3162$, $\tau_\eta = 0.2$ and $\tau_{\sigma^2} = 0.1$.

Figure 5.11 shows the trace plots of the parameter posterior draws from each iteration of the PMCMC algorithm.The histograms of the posterior draws, after a burn-in period of 5000 iterations, are displayed in Figure 5.12.

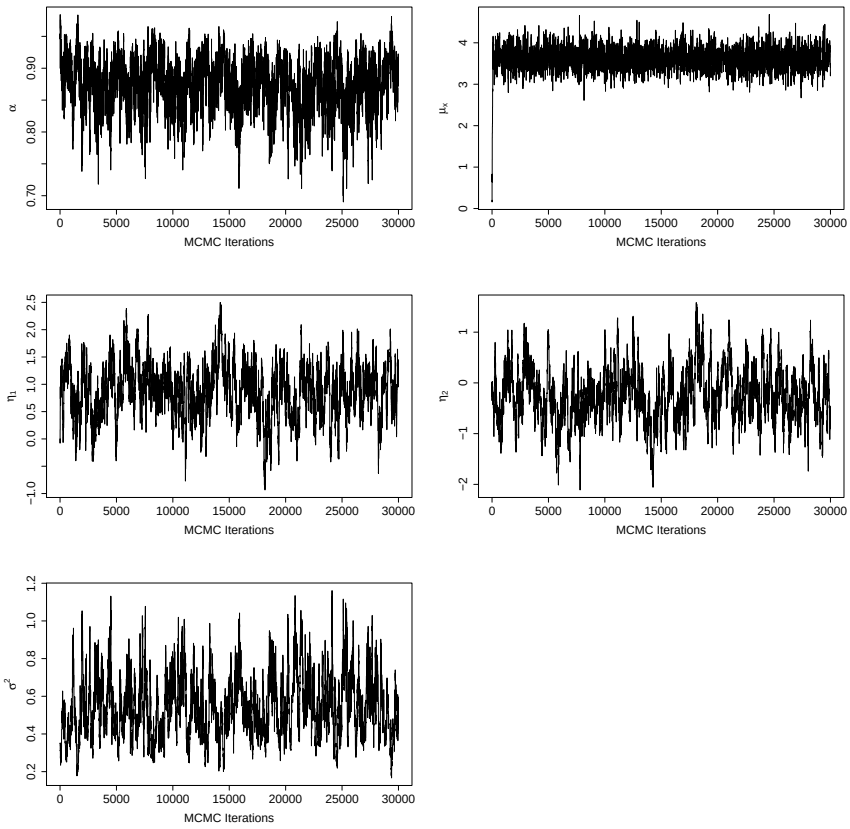

Figure 5.11: SVM with Covariate Effect trace plots of α, μ_x, η_1, η_2 and σ^2 resulting from PMCMC for Bitcoin data.

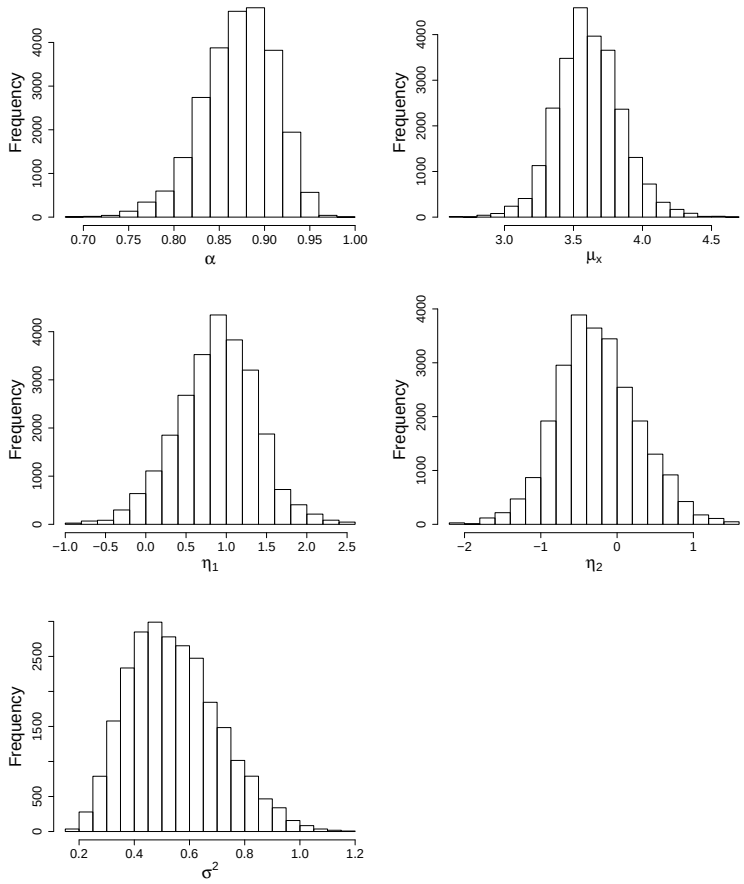

Figure 5.12: SVM with Covariate Effect histograms of α, μ_x, η_1, η_2 and σ^2 resulting from PMCMC for Bitcoin data.

Table 5.4 gives the posterior mean and 95% credible interval resulting from the PMCMC algorithm for each of the model parameters. The acceptance rate was 0.2130 and the $DIC = 4905.48$.

Table 5.4: Summary of SVM with Covariate Effect parameter posterior distributions resulting from PMCMC for Bitcoin data.

Parameter	Mean	95% CI
α	0.872	(0.802, 0.931)
μ_x	3.614	(3.25, 4.01)
η_1	0.879	(0.039, 1.644)
η_2	-0.256	(-1.108, 0.679)
σ^2	0.545	(0.307, 0.837)

5.6 Summary of Bitcoin Data Analysis

In this chapter the stochastic volatility models introduced in Chapter 2 were fit to daily Bitcoin exchange rate data over the period of June 20, 2014 to June 30, 2016. The Particle MCMC algorithm was applied to these models to estimate the parameter values. The Deviance Information Criterion (DIC), outlined in Section 3.6, was calculated for each model and can be used to compare their effectiveness. Table 5.5 gives a summary of the DIC for each model. For the Bitcoin exchange rate data set the most effective model was SVM with Leverage Effect with a DIC of 4903.7.

Table 5.5: A summary of the DIC for each stochastic volatility model that was fit to the Bitcoin exchange rate data set.

Model	DIC
Basic SVM 2	4910.63
SVM with Fat Tail	4914.64
SVM with Leverage Effect	4903.70
SVM with Covariate Effect	4905.48

Similar to the previous chapter, the effectiveness of each model was sensitive to the choice of scale parameters (τ) which had to be specified for the proposal distributions for each model parameter.

Chapter 6

Conclusion and Future work

Stochastic volatility models are useful tools for modeling time series data and are commonly used in financial applications. The motivation for this project was to determine if stochastic volatility models could be used to model the exchange rate of Bitcoin. Chapter 2 described all of the stochastic volatility models that were used in this project. We considered a basic SVM and several extensions including fat tails, leverage, and covariate effects.

The Bayesian approach with the particle Markov chain Monte Carlo (PMCMC) method was employed to estimate the stochastic volatility model parameters. Chapter 3 provided a detailed description of PMCMC as well as the basic MCMC algorithm. First, we introduced the concept of Monte Carlo integration as a simulation technique that uses independent draws from the target distribution to approximate integrals rather than solving them analytically. However, it is not always possible to sample independent draws directly from the target distribution. In this case, we can use Markov chain Monte Carlo to make similar approximations using slightly dependent draws from a Markov chain.

Next, we saw that Importance Sampling is another way to address the problem of not being able to sample independent draws directly from the target distribution. Then by restricting the importance distribution to a certain form that can be written recursively, we can implement sequential importance sampling (SIS) which can be more computationally efficient. A resampling step was then added to the sequential importance sampling algorithm which led to the concept of sequential Monte Carlo. The steps of the SMC algorithm were outlined in detail and we saw how this method provides an estimate of the hidden process X_n. Finally, we described the steps of the particle Markov chain Monte Carlo algorithm which can be used to estimate the SVM parameters. Sequential Monte Carlo is used within the PMCMC algorithm to estimate marginal likelihoods. In this project, the

Metropolis-Hastings method was used for PMCMC which is known as the particle marginal Metropolis-Hastings (PMMH) method.

Chapter 4 contains the results of simulation studies that were conducted to evaluate the performance of the PMCMC method. For each SVM we set the true parameter values and generated a simulated data set. The PMCMC method was used to estimate these model parameters based on the simulated observations $Y_{1:n}$ and the estimates were compared to the true parameter values that we were used to generate the data. We demonstrated that PMCMC can be used to estimate stochastic volatility model parameters, however it was difficult to choose the optimal values for the scale parameters in the parameter proposal distributions.

The results of the Bitcoin data analysis were presented in Chapter 5. In this case, the true SVM parameter values were unknown and the relative change in the exchange rate of Bitcoin were the observations $Y_{1:n}$. The effectiveness of each model was assessed using the deviance information criterion (DIC) and the best model was the SVM with Leverage Effect. Again we found that the effectiveness of each model was extremely sensitive to the choice of scale parameters (τ) which had to be specified for the proposal distributions for each model parameter.

Overall this approach showed some potential for modeling the exchange rate of Bitcoin, however, a method to determine the optimal scale parameters would be required before this approach could contribute to an effective trading strategy. Future research on this topic could focus on the development of a better PMCMC algorithm that is not as difficult to tune. We would like to use more advanced Monte Carlo methods to help improve the mixing of the Markov chain. For example, [12] introduced new latent variables and then used the MCMC moves to update the latent variables and use the SMC to propose new values for the parameters and stochastic process given the latent variables. They considered both particle Gibbs and PMMH.

More complex models could also be explored, such as a stochastic volatility model with both leverage and covariate effects. Further research into additional model covariates would likely help to make this approach more useful for trading Bitcoin. In a complex financial application such as this, there are many factors that must be considered. There are a wide variety of variables that could be related to the price of Bitcoin and this extra information could lead to us being able to more accurately model Bitcoin exchange rates.

Bibliography

[1] Tomohiro Ando. Bayesian inference for nonlinear and non-gaussian stochastic volatility model with leverage effect. *Journal of the Japan Statistical Society*, 36(2):173–197, 2006.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[3] Nando de Freitas Arnaud Doucet and Neil Gordon. An introduction to sequential Monte Carlo methods. pages 3–14, 2001.

[4] Olivier Cappé, Simon J. Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, May 2007.

[5] Bradley P Carlin and Nicholas G Polson. Inference for nonconjugate Bayesian models using the Gibbs sampler. *Canadian Journal of statistics*, 19(4):399–405, 1991.

[6] Joshua CC Chan. The stochastic volatility in mean model with time-varying parameters: An application to inflation modeling. *Journal of Business & Economic Statistics*, (just-accepted), 2015.

[7] Jeffrey Chu, Saralees Nadarajah, and Stephen Chan. Statistical analysis of the exchange rate of Bitcoin. *PLoS ONE*, 10(7), 2015.

[8] Garcia D, Tessone CJ, Mavrodiev P, and Perony N. The digital traces of bubbles: Feedback cycles between socio-economic signals in the bitcoin economy. *Journal of the Royal Society Interface.*, 11(20140623), 2014.

[9] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. 2009.

[10] Bjørn Eraker, Michael Johannes, and Nicholas Polson. The impact of jumps in volatility and returns. *The Journal of Finance*, 58(3):1269–1300, 2003.

[11] Nicholas G Polson Eric Jacquier and Peter E Rossi. Bayesian analysis of stochastic volatility models. *Journal of Business and Economic Statistics*, 12(4), 1994.

[12] Paul Fearnhead and Loukia Meligkotsidou. Augmentation schemes for particle MCMC. *Statistics and Computing*, pages 1–14, 2015.

[13] John Geweke. Bayesian treatment of the independent student-t linear model. *Journal of Applied Econometrics*, 8(S1):S19–S40, 1993.

[14] N. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, April 1993.

[15] Roman Holenstein. *Particle Markov Chain Monte Carlo*. PhD thesis, University of British Columbia, 2009.

[16] Eric Jacquier, Nicholas G Polson, and Peter E Rossi. Models and priors for multivariate stochastic volatility. Technical report, CIRANO, 1995.

[17] Eric Jacquier, Nicholas G Polson, and Peter E Rossi. Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. *Journal of Econometrics*, 122(1):185–212, 2004.

[18] Siem Jan Koopman and Eugenie Hol Uspensky. The stochastic volatility in mean model: empirical evidence from international stock markets. *Journal of applied Econometrics*, 17(6):667–689, 2002.

[19] Noureddine Krichene. Modeling stochastic volatility with application to stock returns. *International Monetary Fund*, (3-125), 2003.

[20] Ladislav Kristoufek. Bitcoin meets google trends and wikipedia: Quantifying the relationship between phenomena of the internet era. *Scientific Reports*, 3(3415), 2013.

[21] Patrick Lam. MCMC methods: Gibbs sampling and the metropolis-hastings algorithm. *Harvard University*.

[22] Fredrik Lindsten, Michael I Jordan, and Thomas B Schön. Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research*, 15(1):2145–2184, 2014.

[23] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[24] Hedibert F Lopes and Nicholas G Polson. Extracting sp500 and NASDAQ volatility: The credit crisis of 2007-2008. *Handbook of Applied Bayesian Analysis*, pages 319–342, 2010.

[25] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[26] Nima Nonejad. Particle Gibbs with ancestor sampling for stochastic volatility models with: heavy tails, in mean effects, leverage, serial dependence and structural breaks. *Studies in Nonlinear Dynamics & Econometrics*, 2014.

[27] Sapuric S and Kokkinaki A. Bitcoin is volatile! isn't that right? *Business Information Systems Workshops, Lecture Notes in Business Information Processing*, pages 255–265, 2014.

[28] David Spiegelhalter, Nicola G Best, Bradley P Carlin, and Angelika van der Linde. Bayesian measures of model complexity and fit (with comments). *Journal of the Royal Statistical Society*, 64(4):583–639, 2002.

[29] Jun Yu and Renate Meyer. Multivariate stochastic volatility models: Bayesian estimation and model comparison. *Econometric Reviews*, 25(2-3):361–384, 2006.